Macro Scheduler

Windows Automation Made Easy

© 2015 MJT Net Ltd

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Web:

http://www.mjtnet.com/

Email:

support@mjtnet.com

Phone:

+44 8700 684 523
+1 360 519 5383

Fax:

+44 709 214 3231
+1 866 788 7502

Table of Contents

Part I Using Macro Scheduler 2

1 Getting Started ... 2
2 Menu Commands ... 3
3 Toolbar ... 4
4 Creating Scripts ... 4
5 Backing up your Scripts and Settings ... 6
6 Using Variables ... 9
7 Code Builder ... 10
8 Code Wizards ... 10
9 Code Snippets ... 11
10 Code Folding and Bookmarks ... 12
11 Scheduling Scripts ... 12
12 Advanced Scheduling ... 15
13 Trigger ... 16
14 AutoLogon ... 17
15 Logging ... 18
16 Encryption ... 18
17 Hot Keys ... 19
18 Groups ... 19
19 Stats ... 20
20 Recording Macros ... 20
21 Playing Scripts And Macros ... 21
22 Creating Desktop Shortcuts ... 22
23 Using the Debugger ... 22
24 Dialog Designer ... 24
25 WebRecorder ... 25
26 Command Line Options ... 26
27 Help and Resources ... 28

Part II Menu Commands 31

1 New Macro ... 31
2 New Macro From Template ... 31
3 Edit Macro ... 31
4 Import Macros ... 31
5 Hide ... 32
6 Exit ... 32
7 Delete ... 32
8 Remove from Group ... 32
9 Rename ... 32

© 2015 MJT Net Ltd

10 Select All ... 32
11 New Group ... 32
12 Delete Group ... 33
13 Group Properties ... 33
14 Sort ... 33
15 Font ... 33
16 Grid Lines ... 34
17 List ... 34
18 Details ... 34
19 Large Buttons ... 34
20 Toolbar ... 34
21 Search Bar ... 34
22 Create Exe ... 34
23 View Log File ... 36
24 Desktop Shortcut ... 36
25 View System Windows ... 36
26 Options ... 37
27 Run Now ... 38
28 Record ... 38
29 Stop ... 38
30 Contents ... 38
31 Check for Update ... 38
32 Support Forums ... 38
33 About ... 39

Part III Scripting Windows For Beginners 41

1 Introduction ... 41
2 Requirements ... 41
3 The Problem ... 41
4 Using Macro Scheduler ... 41
5 First Steps ... 42
6 Building The Script ... 42
Run Notepad ... 44
Wait for the window "Untitled – Notepad" ... 44
Send our lines of text ... 44
Press ALT-FA ... 46
Wait for the window "Save As" ... 46
Send our filename ... 46
The Script So Far ... 48
Printing the Document ... 49
Closing Notepad ... 50
7 The Complete Script ... 50
8 Where To Go From Here ... 52

© 2015 MJT Net Ltd

Part IV Command Reference 54

1 Complex Expressions 54
Arithmetic Operators 54
Logical Operators 55
String Operators 55
Relational Operators 55
Arithmetic Functions 55
String Functions 56
Conditional Functions 57
Examples 57
2 Clipboard Commands 58
GetClipBoard 58
PutClipBoard 58
WaitClipBoard 58
3 Console App Funtions 59
SOWrite 59
SOWriteLn 59
4 Database Commands 59
DBClose 59
DBConnect 60
DBExec 60
DBQuery 61
5 Date/Time Commands 62
DateAdd 62
DateDiff 63
DateLocal 63
DatePart 63
DateStamp 64
Day 64
DayOfWeek 64
GetDate 65
GetTime 66
Hour 66
Min 66
Month 67
Sec 67
TimeAdd 67
TimeDiff 68
TimePart 68
TimeLocal 68
Timer 69
TimeStamp 69
Year 69
6 DDE Commands 70
DDEPoke 70
DDERequest 70
7 Dialogs 70
Dialog Objects 71
File Browse Buttons 71
Button Object Properties 72
Form Properties 73
MainMenu - Menu Builder 76
Label Object Properties 76
Edit Object Properties 78

© 2015 MJT Net Ltd

Memo Object Properties ... 79
CheckBox Object Properties ... 81
ListBox Object Properties ... 83
ComboBox Object Properties ... 85
Image Object Properties ... 87
RadioGroup Object Properties ... 88
ProgressBar Object Properties ... 90
PageControl Object Properties ... 91
TabSheet Object Properties ... 93
Panel Object Properties ... 94
Splitter Object Properties ... 96
StatusBar Object Properties ... 97
StringGrid Object Properties ... 98
MenuItem Object Properties ... 100
HTMLViewer Object Properties ... 101
Dialog Functions ... 101
CloseDialog ... 102
Dialog ... 102
EndDialog ... 102
GetDialogAction ... 103
PrintDialog ... 103
ResetDialogAction ... 103
SetDialogObjectColor ... 103
SetDialogObjectFont ... 104
SetDialogObjectFocus ... 104
SetDialogObjectVisible ... 105
Show ... 105
GetDialogProperty ... 105
SetDialogProperty ... 106
AddDialogHandler ... 106
Dialog Examples ... 107
Modal vs Non-Modal Dialogs ... 107
8 Excel Functions ... 109
XLAddSheet ... 109
XLCreate ... 109
XLDelCol ... 110
XLDelRow ... 110
XLDelSheet ... 111
XLGetCell ... 111
XLGetSheetDims ... 112
XLOpen ... 112
XLQuit ... 113
XLRun ... 113
XLSave ... 113
XLSetCell ... 114
XLSheetToArray ... 114
9 File Handling ... 115
AppendFile ... 115
ChangeDirectory ... 116
CopyFile ... 116
CopyFolder ... 116
CountDirs ... 117
CountFiles ... 118
CreateDir ... 118
CSVFileToArray ... 118
DeleteFile ... 119
DeleteFolder ... 120

© 2015 MJT Net Ltd

EditIniFile ... 120
FileDate ... 121
FileSize ... 121
FileTime ... 121
GetFileList ... 122
GetNewestFile ... 122
GetOldestFile ... 123
IfDirExists ... 123
IfFileChanged ... 124
IfFileExists ... 124
IfNotDirExists ... 125
IfNotFileChanged ... 125
IfNotFileExists ... 126
MoveFile ... 126
ReadFile ... 127
ReadIniFile ... 127
ReadLn ... 127
RenameFile ... 128
WriteLn ... 128
ZipAddFiles ... 129
ZipExtractFiles ... 129

10 FTP/HTTP/Telnet/Email Commands ... 130

FTPDelFile ... 130
FTPGetDirList ... 131
FTPGetFile ... 132
FTPMakeDir ... 133
FTPPutFile ... 134
FTPRemoveDir ... 135
FTPRenameFile ... 136
HTTPRequest ... 137
IsConnectedToInternet ... 138
IEGetTags ... 139
IEGetTagsByAttrib ... 140
IETagEvent ... 140
IETagEventByAttrib ... 141
IEWait ... 142
RetrievePOP3 ... 142
SMTPSendMail ... 143
TelnetClose ... 144
TelnetConnect ... 144
TelnetSend ... 145
TelnetWaitFor ... 145

11 Conditional Commands (If...) ... 146

If ... 146
IfDirExists ... 147
IfFileChanged ... 148
IfFileExists ... 148
IfNot ... 149
IfNotDirExists ... 149
IfNotFileChanged ... 150
IfNotFileExists ... 150
IfNotWindowOpen ... 151
IfWindowOpen ... 151

12 Image Recognition ... 152

CompareBitmaps ... 152
FindImagePos ... 152
GetPixelColor ... 154

© 2015 MJT Net Ltd

GetRectCheckSum ... 155
GetScreenRes ... 155
ScreenCapture ... 156
WaitPixelColor ... 156
WaitRectChanged ... 157
WaitScreenImage ... 157
13 Keyboard Commands ... 158
Ascii ... 158
CapsOff ... 158
CapsOn ... 158
NumOff ... 159
NumOn ... 159
ObjectSendKeys ... 159
ObjectSendText ... 159
Press ... 160
Release ... 161
ScrollOff ... 162
ScrollOn ... 162
SendText ... 162
WaitKeyDown ... 163
14 Loops and Branching ... 163
EndWhile ... 163
Gosub ... 164
Goto ... 165
Label ... 165
Repeat ... 166
SkipLabel ... 166
Until ... 167
While ... 167
15 Messages ... 168
Ask ... 168
Input ... 168
Message ... 169
MessageModal ... 170
16 Miscellaneous ... 170
AddTrayHandler ... 170
AddTrayIcon ... 171
ArrayCount ... 171
ArrayDim ... 172
ArraySort ... 172
ColorToRGB ... 172
DelArray ... 173
DelTrayIcon ... 173
DelVariable ... 173
BlockInput ... 174
ExportData ... 174
GetEnvVar ... 174
KillProcess ... 175
LibFree ... 175
LibFunc ... 175
LibFuncW ... 177
LibLoad ... 177
ModTrayIcon ... 178
PlayWav ... 178
PopupMenu ... 178
ProcessExists ... 179

© 2015 MJT Net Ltd

PyExec ... 179
Random ... 180
RGB ... 181
SetEnvVar ... 181
WaitCursorChanged ... 181
WaitProcessExists ... 182
WaitProcessTerminated ... 182
Wow64DisableRedirection ... 182
Wow64EnableRedirection ... 183
17 Mouse Commands ... 183
GetCaretPos ... 183
GetCursorPos ... 184
LClick ... 184
LDblClick ... 184
LDown ... 185
LUp ... 185
MClick ... 185
MDblClick ... 185
MDown ... 186
MouseMove ... 186
MouseMoveRel ... 186
MouseOver ... 187
MUp ... 187
RClick ... 188
RDblClick ... 188
RDown ... 188
RUp ... 188
Toolbar ... 189
18 Numeric Functions ... 189
Add ... 189
Let ... 190
SetRoundMode ... 191
Sub ... 191
19 Registry Functions ... 192
RegistryDelKey ... 192
RegistryDelVal ... 192
RegistryEnumKeys ... 192
RegistryEnumVals ... 193
RegistryReadKey ... 193
RegistryWriteKey ... 193
20 Running Programs/Files ... 194
ExecuteFile ... 194
RunProgram ... 194
21 Script Control ... 196
Sub Routines ... 196
Gosub ... 196
SRT ... 197
END ... 198
SkipLabel ... 198
Assigned ... 199
EndWhile ... 199
Exit ... 199
Goto ... 200
If ... 200
IfNot ... 201
Include ... 202

© 2015 MJT Net Ltd

IncludeFromVar ... 202
Label ... 202
Let ... 203
Macro ... 204
OnEvent ... 204
Remark ... 208
Repeat ... 208
Until ... 209
Wait ... 209
WaitReady ... 209
While ... 210
22 String Handling ... 210
AESEncrypt ... 210
Base64 ... 211
ConCat ... 212
Crypt ... 212
ExtractFileExt ... 213
ExtractFileName ... 213
ExtractFilePath ... 213
Format ... 214
Hash ... 215
HTMLDecode ... 215
HTMLEncode ... 215
JSONParse ... 215
LabelToVar ... 216
Length ... 217
LowerCase ... 217
LTrim ... 217
MidStr ... 217
Position ... 218
RegEx ... 218
RTrim ... 219
Separate ... 219
StringReplace ... 219
Trim ... 220
UpperCase ... 220
XMLParse ... 220
23 Text Capture ... 221
GetControlText ... 222
GetTextAtPoint ... 222
GetTextInit ... 223
GetTextInRect ... 223
GetTextPos ... 224
GetTextReset ... 224
GetWindowText ... 224
GetWindowTextEx ... 225
WaitScreenText ... 225
24 VBScript Commands ... 226
VBEND ... 226
VBEval ... 226
VBSTART ... 228
VBRun ... 228
25 WebRecorder Functions ... 229
IEClickTag ... 229
IEContainsText ... 230
IECreate ... 230

© 2015 MJT Net Ltd

IEExtractTable ... 231
IEExtractTableByName ... 231
IEExtractTag ... 232
IEExtractTagByAttrib ... 232
IEExtractTagByName ... 233
IEFormFill ... 233
IEFormSubmit ... 234
IEGetFromURL ... 234
IEGetHWND ... 235
IEGetTagPos ... 235
IEGetURL ... 235
IEGoBack ... 236
IEGoForward ... 236
IENavigate ... 236
IEOnDownload ... 237
IEQuit ... 237
IERefresh ... 238
IESetTimeout ... 238
IEShowIE ... 238
IEWait ... 239
IEWaitDocumentComplete ... 239
IEWaitFileDownload ... 239
IEWaitForText ... 240
IEWaitTimeout ... 240
IEWaitNew ... 241

26 Window Functions ... 241

CloseWindow ... 241
FindWindowWithText ... 242
GetActiveWindow ... 242
GetWindowChildList ... 243
GetWindowHandle ... 243
GetWindowList ... 243
GetWindowNames ... 244
GetWindowParent ... 244
GetWindowPos ... 245
GetWindowProcess ... 245
GetWindowSize ... 246
GetWindowText ... 246
IfWindowOpen ... 247
IfNotWindowOpen ... 248
MoveWindow ... 248
ResizeWindow ... 249
SetFocus ... 249
ShutDownWindows ... 250
WaitWindowChanged ... 251
WaitWindowClosed ... 251
WaitWindowFocused ... 252
WaitWindowOpen ... 253
WindowAction ... 254

27 Window Objects ... 255

FindObject ... 255
GetCheckBox ... 255
GetControlText ... 256
GetFocusedObject ... 256
GetListItem ... 257
GetTreeNode ... 258
PushButton ... 259

© 2015 MJT Net Ltd

SelectMenu ... 260
SetCheckBox ... 260
SetControlText ... 261
SetObjectText ... 261
UIAccessibleList ... 261
UIClick ... 262
UIFocus ... 262
UIGetValue ... 263
UIPos ... 263
UISelect ... 263
UISetValue ... 264
28 Using Variables ... 264
User Defined Variables ... 264
Arrays ... 265
Scope ... 266
Explicit Variable Resolution - VAREXPLICIT ... 267
Ignoring Spaces - IGNORESPACES ... 268
System Variables ... 269
Variable Explorer ... 272
29 Error Handling ... 272
Index 273

© 2015 MJT Net Ltd

Using Macro Scheduler

Part I

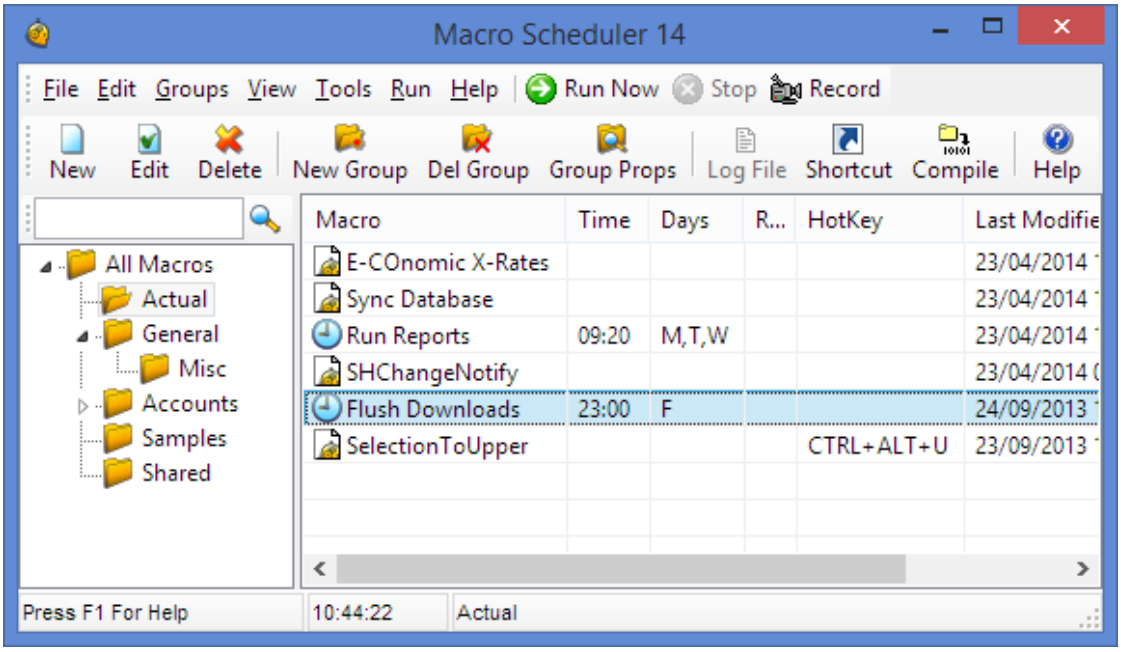
1 Using Macro Scheduler

1.1 Getting Started

When Macro Scheduler starts it places its icon in the system tray (next to the clock). The first time Macro Scheduler is run it starts minimized. To view Macro Scheduler click on this icon (or double click if you are still stuck using XP), or right click on the icon to display the pop up menu and select 'Show'.

Note: as of Macro Scheduler 14.1.03 the icon you will see in the tray is:
Previous versions of Macro Scheduler used:

If you like you can prevent Macro Scheduler from starting minimized. To do this select Tools/Options and uncheck 'Start Minimized'. See Options 37.



This is the main control center for Macro Scheduler. From here macros can be started, recorded, deleted and edited.

For further information on creating macros see Creating Scripts 4 and Recording Macros 20

Or see Menu Commands 3 or Toolbars 4 for an explanation of the menu options and toolbar buttons.

Access to the functions is through the toolbar buttons, the system menu or by double clicking on the appropriate macro, or right clicking on it to display a pop up menu.

The search bar can be used to locate macros. Macro Scheduler searches the content of macros as well as the macro names. Results will include macros which contain all of the words specified in the search, whatever their order. To search for a specific phrase include it in double quotes.

© 2015 MJT Net Ltd

1.2 Menu Commands

File

New Macro 31
New Macro From Template 31
Edit Macro 31
Import Macros 31
Hide 32
Exit 32

Edit

Delete 32
Remove from Group 32
Rename 32
Select All 32

Groups

New Group 32
Delete Group 33
Group Properties 33

View

Sort 33
Font 33
Grid Lines 34
List 34
Details 34
Large Buttons 34
Toolbar 34
Search Bar 34

Tools

View Log File 36
Create Exe 34
Desktop Shortcut 36
View System Windows 36
Options 37

Run

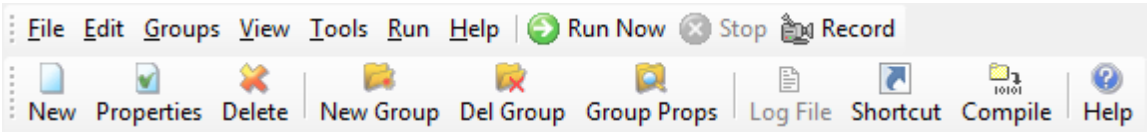
Run Now 38
Record 38
Stop 38

Help

Contents 38
Check for Update 38
Support Forums 38
About 39

© 2015 MJT Net Ltd

1.3 Toolbar



The top row of the toolbar contains the menu and buttons to Run 38, Stop 38 and Record 38 macros.

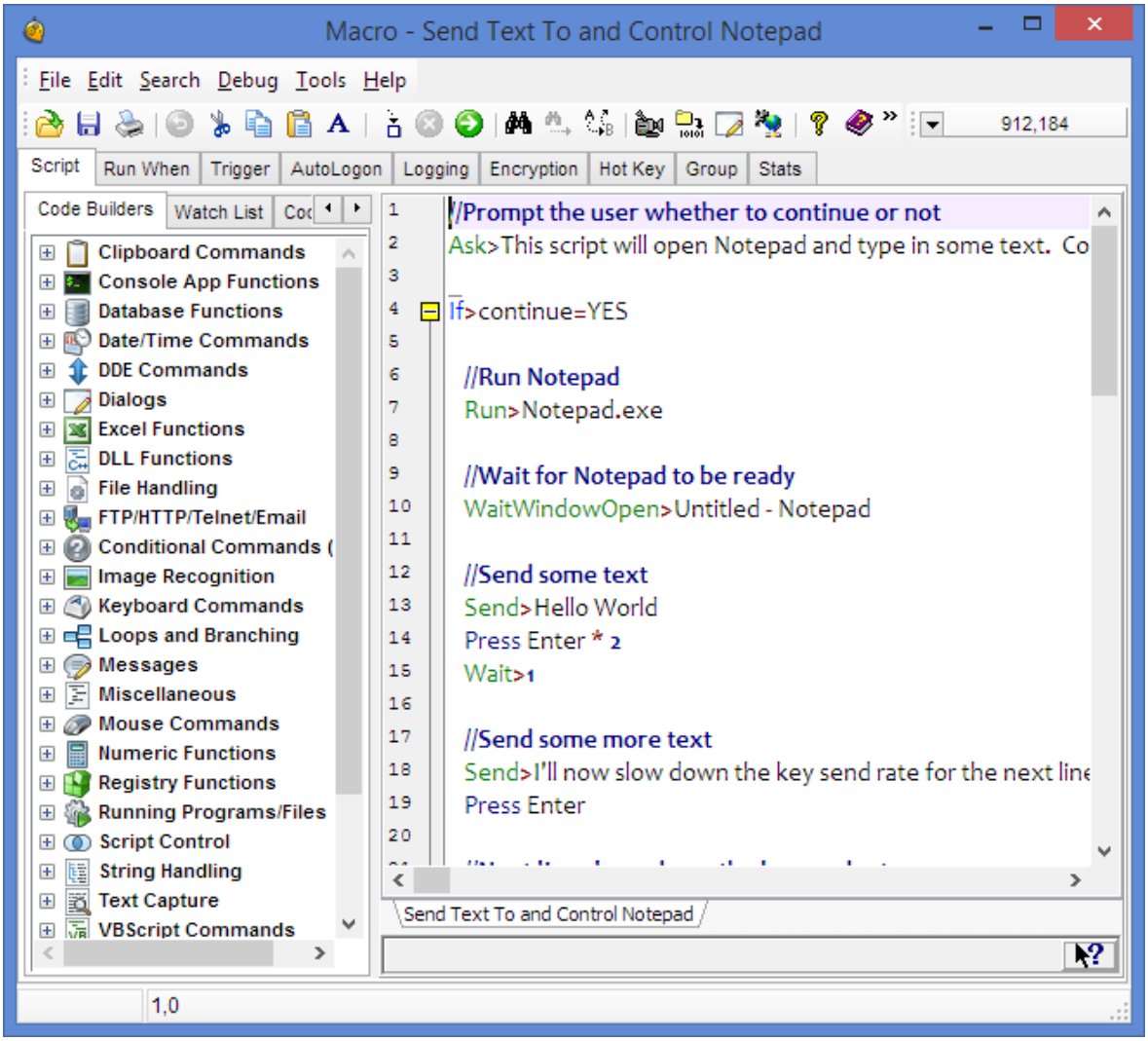
Buttons from Left to Right on the second line are:

New Macro 31, Macro Properties 31, Delete Macro 32, New Group 32, Delete Group 33, Group Properties 33, View LogFile 36, Desktop Shortcut 36, Create Exe 34, Help 38

1.4 Creating Scripts

To create a new script select 'File/New Macro', or click the 'New' button on the Toolbar 4. To Edit an existing script, select it from the list by clicking on it and then select 'File/Macro Properties' or double click on it. Having done this you will be presented with the following window:

© 2015 MJT Net Ltd



In this example an existing script was selected to be edited. If 'New' had been pressed then this form would appear blank.

Code can be inserted and edited directly into the script editor. Use the Code Builders 10 and Wizards 10 found in the left hand Code Builders pane to help locate functions and insert code.

Pressing F1 in the editor will present the help topic for the highlighted function or, if no text is highlighted, the function in the current line. To test the script click the 'Run' button. Note that when running a script from within the editor you are actually running in debug mode. If you have a log file enabled it will not be updated while running in debug mode.

To save the macro press the Save toolbar button. If you have not yet given the macro a name you will be asked for one. Press Save to save any changes made to the macro script or properties.

Save As will allow you to save to a new macro or script file and context will then change to that script/macro. In contrast Export will simply save a copy of the script file but you will continue working on the current macro.

The script editor provides a full featured programmer's editor with code folding, bookmarks, the Debugger 22 and other script building options including the Dialog Designer 24 and Code Snippets. See also Code Folding and Bookmarks 12. You can also set the script backup options 6 from the editor's tools menu.

© 2015 MJT Net Ltd

The Import button allows you to load in a script that has already been created. This is useful if you have a number of Macro Scheduler installations and want to make use of a script created on a different PC for example.

The numbers at the top right of this window show your mouse cursor position. Use these to determine the correct parameters when using the MouseMove command. The button to the left of the numbers reveals a drop down menu which allows you to toggle between absolute coordinates and relative coordinates. When set to relative, the numbers show you the relative coordinates to the window the cursor moves over. The other option allows you to attach a small tag to the mouse cursor that shows the coordinates and follows the cursor around. This means that if the script window becomes concealed by another application you can still see the cursor position. If you then press both left and right mouse buttons the content of the cursor monitor window will be copied to the clipboard, making it easy to paste into your code. The last option of the drop down menu is used to add the pixel colour to the display as well as the cursor position. This is helpful for the WaitPixelColor 156 command.

The editor is now "multi-tabbed". That is, you can have more than one file open in the editor at once. The tabs are positioned at the bottom of the editor. To open another file in a new tab select File/ Open in New Tab. You can also right click on a filename that appears anywhere within the editor and select "Open in New Tab" to open it in a new tab.

For help with other tabs of this window see: Scheduling Scripts 12 (Run When), Trigger 16, AutoLogon 17, Logging 18, Encryption 18, Hot Keys 19, Groups 19

To get help with the MacroScript language see: Scripting Windows for Beginners 41, Command Reference, Using Variables 264, Complex Expressions 54

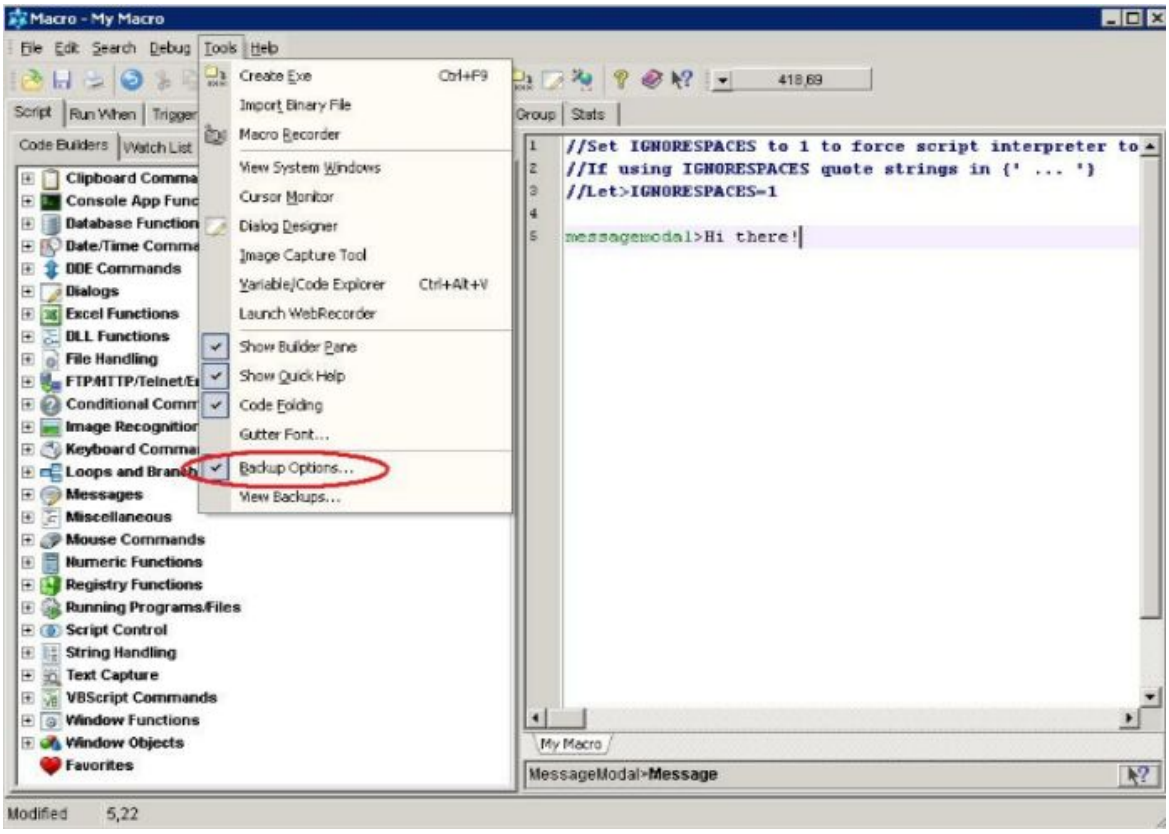
1.5 Backing up your Scripts and Settings

Macro Scheduler can automatically backup your last few saves for each Macro. In v14, this is turned on by default, and will backup your Macro every time you Save.

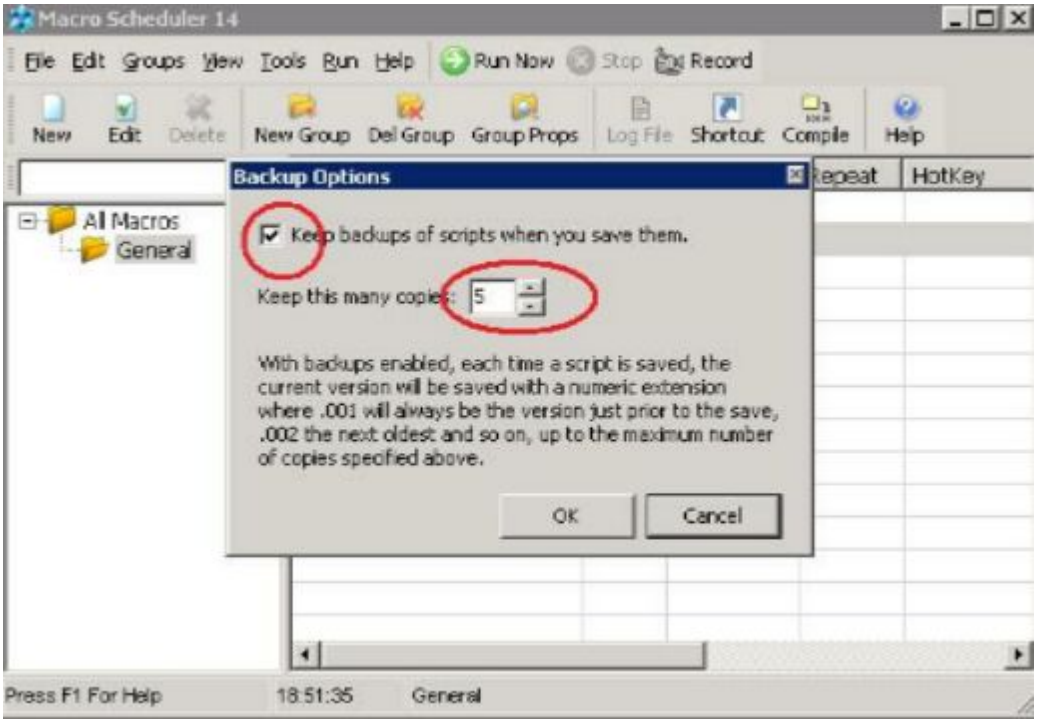
To view and change Backup Options

From the *Script Editor*, click *Tools*, *Backup Options*

© 2015 MJT Net Ltd



Enable/disable Macro backup, and select how many copies to keep.

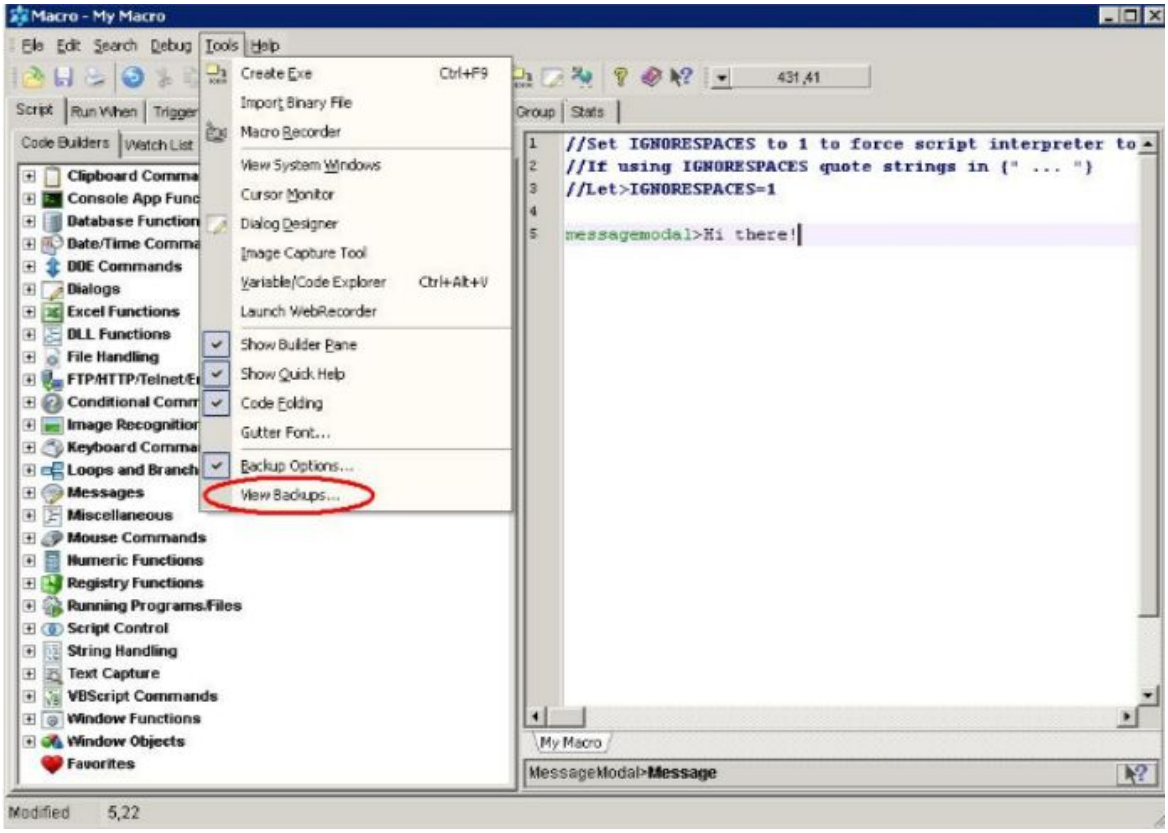

Macro Scheduler 14
File Edit Groups View Tools Run Help Run Now Stop Record
New Edit Delete New Group Del Group Group Props Log File Shortcut Compile Help
Backup Options
All Macros
General
Keep backups of scripts when you save them.
Keep this many copies: 5
With backups enabled, each time a script is saved, the current version will be saved with a numeric extension where .001 will always be the version just prior to the save, .002 the next oldest and so on, up to the maximum number of copies specified above.
OK
Cancel
HotKey
Press F1 For Help
18:51:35
General

To access your backups

Backups will be created each time a Macro is saved. This creates Macro.001, Macro.002, and so

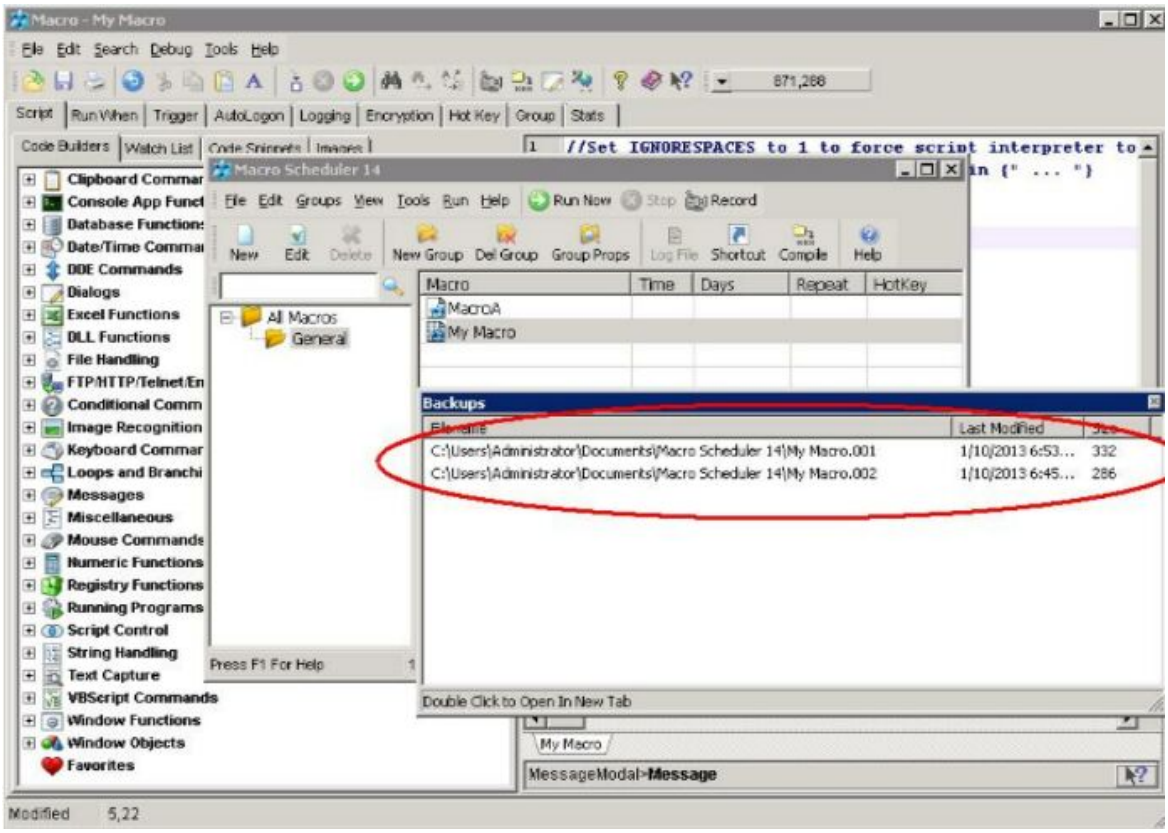
© 2015 MJT Net Ltd

on.
From the *Script Editor*, click *Tools*, *View Backups*



A list of backups will be presented. Double click a backup in order to open it.

© 2015 MJT Net Ltd


Macro - My Macro
File Edit Search Debug Tools Help
Script Run When Trigger AutoLogon Logging Encryption Hot Key Group Stats
Code Builders Watch List
Macro Scheduler 14
File Edit Groups View Tools Run Help Run Now Stop Record
New Edit Delete New Group Del Group Group Props Log File Shortcut Compile Help
Macro Time Days Repeat HotKey
MacroA
My Macro
All Macros
General
Backups
Filename Last Modified
C:\Users\Administrator\Documents\Macro Scheduler 14\My Macro.001 1/10/2013 6:53... 332
C:\Users\Administrator\Documents\Macro Scheduler 14\My Macro.002 1/10/2013 6:45... 286
Press F1 For Help
Double Click to Open In New Tab
My Macro
MessageModal>Message
Modified 5,22

To open a backup in a new editor tab double click it or right click and select "Open in Editor Tab". To open the folder in Explorer right click and select "View Folder".

Restoring Backups

The backup files are stored in the same folder as the macro's script file. For macros set up inside Macro Scheduler this is the folder associated with the macro's group. You can see the path in the View Backups dialog under Filename (see above). To quickly open the folder in Explorer right click on a backup and click "View Folder" from the pop up menu.

To restore a backup simply rename it by changing the extension to .scp. You may need to rename or delete the main script file first.

Backing up Settings and Schedules

Macro Scheduler automatically backs up it's three data files every time you start it up. It will keep up to three copies of these files using numbered extensions .001 to .003. The files are macros12.dat, groups12.dat and groups12.ini. Should you experience a problem with groups or schedules or lose your macro settings try restoring from one of these backups. These will be stored in the main Macro Scheduler data folder which is usually under My Documents. If you're not sure of the location look under Tools/Options and then click "Change Macro/Group Settings Path (Advanced)" and you will see the current location at the top.

1.6 Using Variables

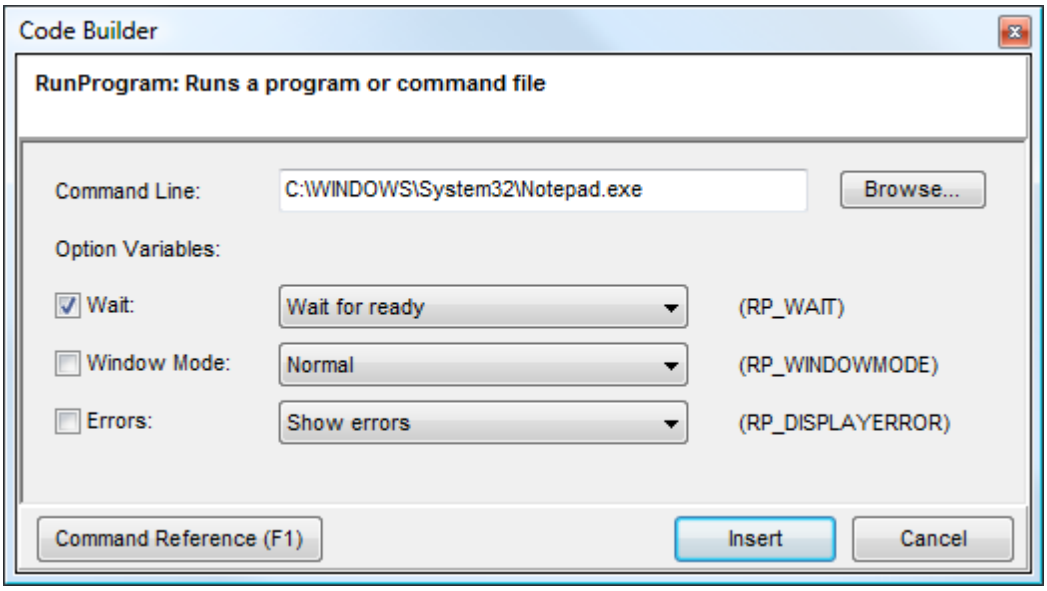
Please see the following topics under the Command Reference chapter:

User Defined Variables 264

© 2015 MJT Net Ltd

Arrays 265
Explicit Variable Resolution - VAREXPLICIT 267
Ignoring Spaces - IGNORESPACES 268
System Variables 269
Variable Explorer 272
Debugger - Watch List 22

1.7 Code Builder



Code Builders are available for each command under the Script tab of the Macro Properties 4 window. The code builder tree organises all the script commands into convenient categories, and in the Script Editor also allows commands to be added and removed from a Favorites category.

Expand a category to see the list of commands. To insert the selected command into the editor, double click the command. The chosen script command will appear in the Code Builder dialog where you can enter the parameter values and options required for that command. Click Insert to insert the code into the editor.

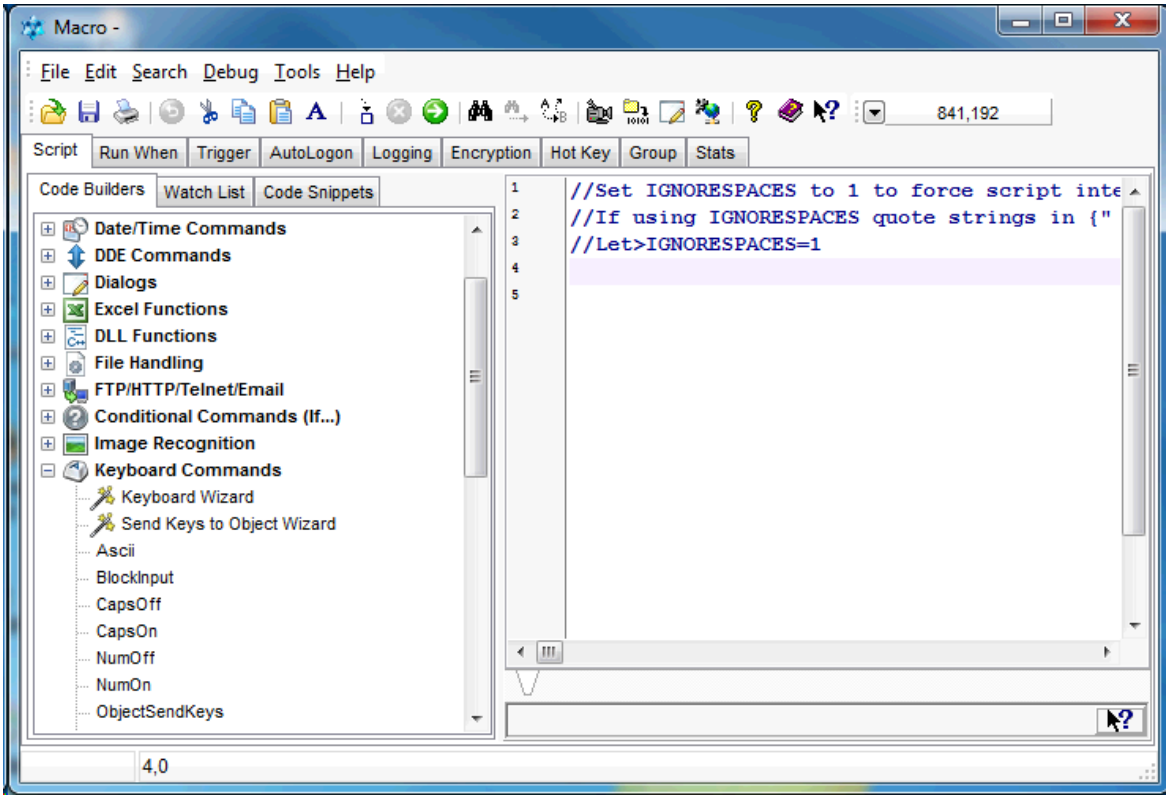
In the Script Editor you can right click on a command and add it to your favorites category using the pop up menu options.

To get help on the command press F1.

1.8 Code Wizards

Several Wizards exist to simplify the creation of code. Wizards can be activated from the Code Builder tree in the editor:

© 2015 MJT Net Ltd



The following wizards exist:

Image Recognition Wizard
Keyboard Wizard
Send Keys to Object Wizard
Mouse Action Wizard
Text Capture Wizard

Each should be self explanatory, but all have links to online videos demonstrating their use.

1.9 Code Snippets

With Code Snippets you can store pieces of frequently used code which can quickly be inserted into a script as you are editing it.

To view Code Snippets click the Code Snippets tab in the left hand pane (the Builder Pane) of the Editor 4.

Right click on the Code Snippets window to reveal the popup menu with options to create categories, create a new snippet, edit, delete and rename snippets. Insert a snippet into your script code by selecting the Insert Snippet option from the popup menu or by double clicking the snippet.

Before a snippet can be created you must first create a category. Once a category has been created you can right click on it and select New Snippet. In the box that appears, enter a name for the snippet and then enter the snippet code. Enter or paste the code into the snippet box and click Save.

Once you have created a snippet it will be available in the editor for you to insert into any script.

© 2015 MJT Net Ltd

1.10 Code Folding and Bookmarks

The Macro Scheduler script editor supports various advanced editing features such as Code Folding and Bookmarks.

Code Folding

Recognised code blocks marked by the editor with a vertical line in the margin showing their extent, and a collapse/expand icon at the first line. You can collapse or expand the code block by clicking on this icon. Collapsing a block of code hides it temporarily from view.

You can also fold any code of your choice by placing the text CODEBLOCK and ENDCODEBLOCK before and after the code. E.g.:

```
CODEBLOCK
Run>Notepad.exe
WaitWindowOpen>Notepad.exe
Send>Hello World
ENDCODEBLOCK
```

Bookmarks

Bookmarks allow you to mark a line so that you can get back to it quickly. Right click on a line and select Create Bookmark. You will be prompted for the bookmark text to use and the bookmark will be created. An icon will appear in the gutter marking the code and the line will be highlighted. You can add as many bookmarks as you wish. To quickly get back to a line right click to invoke the context menu, or select the m,main Search menu and click Goto Bookmark. Choose the bookmark from the list.

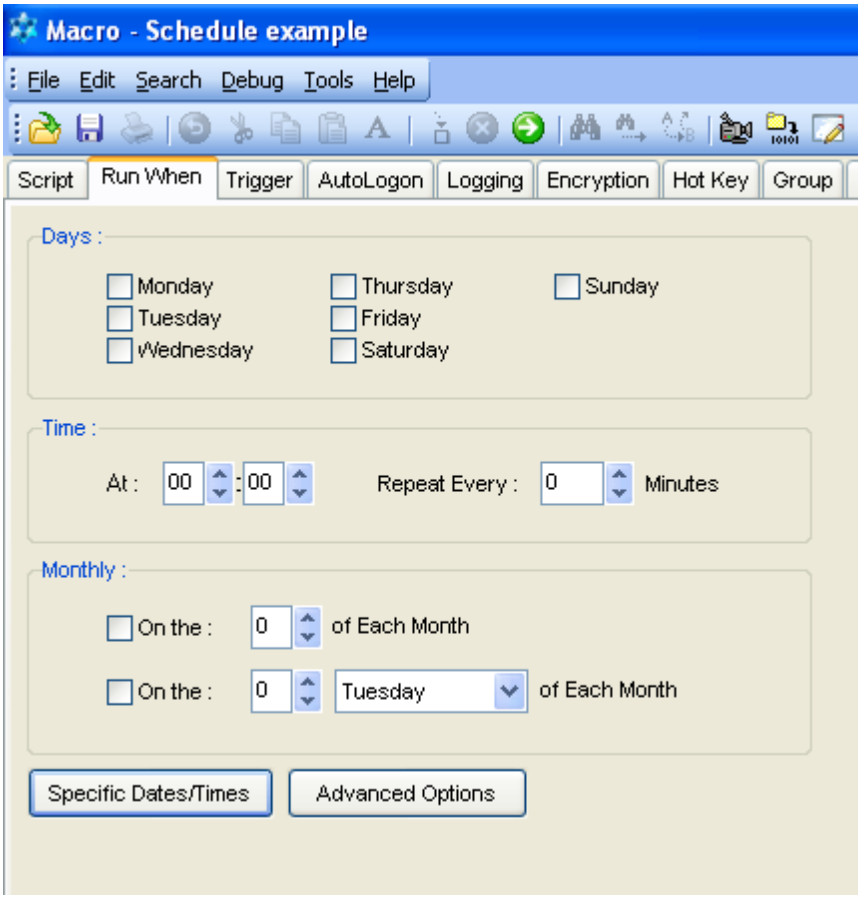
Bookmarks are persistent. When you save your script the bookmarks are saved with it so that when the script is later reopened the bookmarks are reinstated.

1.11 Scheduling Scripts

Once you have created your macro you will probably want to execute it. Macros can be run at any time from the main window, from Windows shortcuts, from the command line or to a specified schedule.

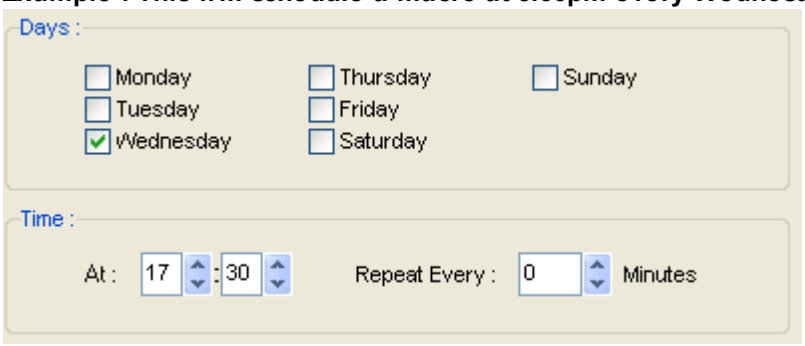
To set up a schedule, select the appropriate script and choose to edit it to invoke the Macro Properties Form 4. Then select the tab marked 'Run When' to display the following options.

© 2015 MJT Net Ltd



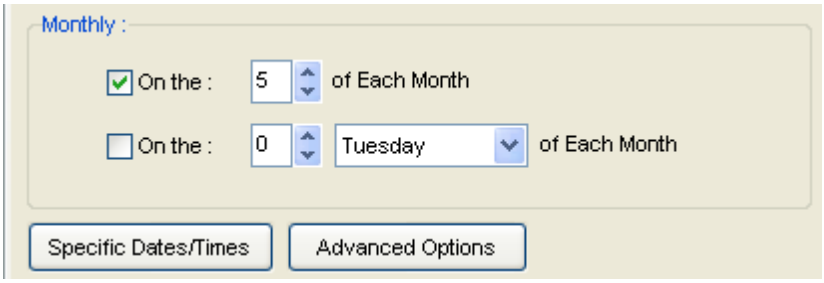
Mark off the days on which you want the macro to run and enter a time. The time must be entered in 24 hour notation. If you want the macro to be repeated enter an appropriate value in the 'repeat every' box. If you don't want it repeated simply leave this set to zero. The repeat every box allows you to specify up to 1440 minutes (a full day).

Example : This will schedule a macro at 5.30pm every Wednesday.



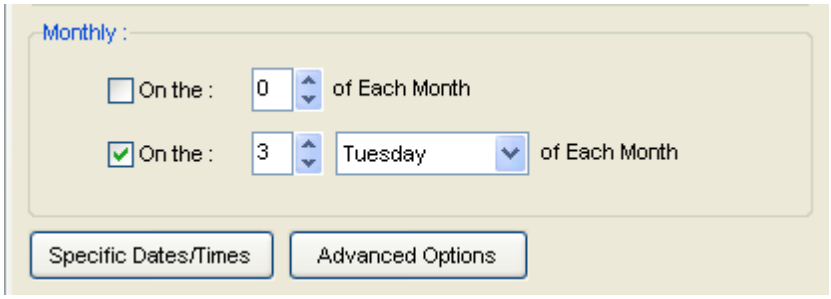
To make the macro run on a monthly basis check the monthly box and enter the day of the month on which you want the macro invoked. Filling in the *time* section will allow you to specify what time the macro will run, on those matching days.

© 2015 MJT Net Ltd

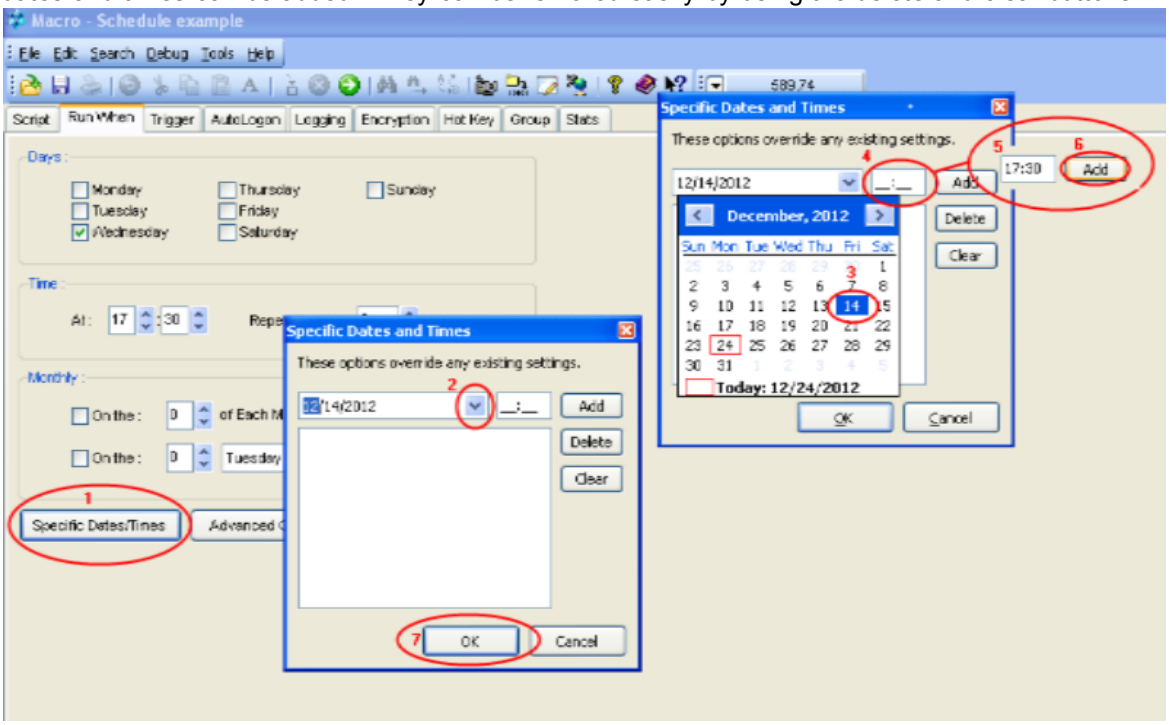


If you choose to run a macro on the 5th of the month and also check the Friday box, the macro will run every Friday AND on the 5th of the month regardless of what day the 5th is.

To run a macro on the 3rd Tuesday of every month:



To set a macro to run at a specific date and time, simply follow the seven steps below. Multiple dates and times can be added. They can be removed easily by using the delete and clear buttons.



In the 'Advanced' scheduling settings you can specify what should happen when Macro Scheduler restarts. This can be useful if you have Macro Scheduler start every time you reboot your computer. For instance, you can set Macro Scheduler to continue to repeat when the computer restarts. You can also determine what should happen if a schedule was missed while the computer was turned off,

© 2015 MJT Net Ltd

and you can specify a window title so that the macro starts when that window appears. See Advanced Scheduling Options 15.

If you have created a schedule for a macro, but wish to temporarily stop scheduling without changing the schedule details, you can do so from the pop up menu of the script list on the main Macro Scheduler window.

A Note About Screensavers

Screensavers usually stop successful detection of other windows. Consequently if a script that needs to setfocus or wait for windows to appear is run while a screensaver is active it may not work correctly.

To get round this Macro Scheduler temporarily disables screensaving just before it runs a script and re-enables screensaving when the script completes. It also attempts to determine if a screensaver is currently active and if so closes it down. However, there are many different implementations of screensavers which operate in different ways, making their detection and close down a rather unreliable process. To try to ensure that Macro Scheduler is successful in closing an active screensaver it briefly moves the mouse back and forth before running the script.

This should work in most cases. However, if you find that Macro Scheduler is unable to stop your screensaver, then the most reliable method of making sure that a scheduled macro runs properly is to simply disable screensaving altogether.

You can also use AutoLogon 17 to unlock a locking screensaver.

1.12 Advanced Scheduling

The Advanced Schedule options dialog is accessed from the Run When 12 tab of the Macro Properties 4 window. Options are:

When Repeating
These options are used when you have set a repeat interval on the main schedule. The default option is repeat continuously. What this means is that even after the system restarts the schedule will continue to repeat (as long as the current day is a day chosen in the main schedule). The second option will tell Macro Scheduler not to keep repeating continuously, but only to repeat until the specified time. At that point the original schedule time is reset to the original start time.

Recovery
Here there are three options. The default is to do nothing except run the macro at the scheduled start time, as normal. If the second option is set, then when Macro Scheduler restarts it checks to see if a macro should have run on that day. If a macro was supposed to run on that day and at a time prior to the system restarting, the macro will be run immediately. The third option will simply allow you to run the macro on startup regardless. Note that the third option will not effect the schedule time in any way - it simply runs the macro.

Scheduler Service
If you have enabled the AutoLogon service in options 37 to schedule your macros when Windows is locked/logged out you can determine whether this macro should be started by the AutoLogon service here. Uncheck Allow Scheduler Service to Run this Macro if you do not want the AutoLogon service to start this macro when Windows is locked/logged out. This option has no effect if the AutoLogon service is not enabled in Options 37.

© 2015 MJT Net Ltd

1.13 Trigger

Under the Trigger tab of the Macro Properties dialog you can specify a condition which should trigger the macro. Trigger types include

Window Event
- Window open
- Window closed

File Event:
- File exists
- File does not exist

Folder Event
- Folder exists
- Folder does not exist
- New file in folder
- File deleted from folder

Custom Trigger
- Allows you to create your own trigger using code by pointing to a script file. See below for details.

Macro Scheduler needs to be running for these triggers to be actioned.

More triggers can be implemented by building them into the script itself. Events and conditions can be coded into continuously looping macros so that they themselves detect conditions and act on them - e.g. screen changes, window changes, object caption changes, event log entries, ini file values, registry values, and almost anything you can think of.

Custom Triggers

Custom triggers allow you to define any kind of trigger imaginable by coding the trigger check yourself using MacroScript code. There are some rules:

A trigger script must have two subroutines as follows:

Trigger: This subroutine is executed periodically (less than once a second) by the scheduler and is used to determine whether or not the macro should fire. Set MACRO_RESULT to "TRUE" (upper case) to cause the trigger to fire (i.e. to run the macro). Setting MACRO_RESULT to anything else (or not setting it) will do nothing. So, the macro will not run until MACRO_RESULT is "TRUE" (upper case). The subroutine name is case sensitive.

Reset: Once Trigger returns "TRUE" the macro will run and thereupon the scheduler will no longer run the Trigger routine and instead will keep running Reset until it returns "TRUE" (upper case). Once Reset returns TRUE the scheduler will again run Trigger. Note, the subroutine name is case sensitive.

Since Trigger routines are run frequently on a very tight interval they should be kept as small as possible and should not be long running. Don't make Trigger scripts that perform too much complicated processing or include delays as this could cause resource issues.

Here's a small example trigger script which implements a simple Window Open trigger:

SRT>Trigger

© 2015 MJT Net Ltd

```
  IfWindowOpen>Calculator
    Let>MACRO_RESULT=TRUE
  Else
    Let>MACRO_RESULT=FALSE
  Endif
END>Trigger

SRT>Reset
  IfWindowOpen>Calculator
    Let>MACRO_RESULT=FALSE
  Else
    Let>MACRO_RESULT=TRUE
  Endif
END>Reset
```

1.14 AutoLogon

AutoLogon can be used to automatically unlock or log on the workstation if it is locked when the macro is scheduled. When the macro has finished running the workstation is locked again.

AutoLogon is only supported on Windows Vista, Windows 7 and Windows Server 2008 and upwards. It is no longer supported on XP and below.

Unlock Computer To Run This Script
Check "Unlock Computer To Run This Script" if you want Macro Scheduler to automatically unlock/ log on to run the script if the workstation is locked at the time. Enter a valid Windows username and password.

The AutoLogon settings relate only to the macro they are set for and effect what happens when the script is scheduled to run at a time when the workstation is locked or logged out.

AutoLogon is only enabled if you are running a supporting operating system (see above) AND the Macro Scheduler Service is installed and running. The Macro Scheduler Service should have been installed and started during installation of Macro Scheduler. If AutoLogon is not enabled check that the service is installed and running by going to Control Panel -> System -> Administrative Tools -> Services. If it is installed but not running you can start it here. If it is not in the list then Macro Scheduler needs reinstalling.

AutoLogon works with two scenarios and each has some requirements:

Unlocking a Locked Workstation

To unlock a locked workstation Macro Scheduler needs to remain running. With AutoLogon enabled you can leave Macro Scheduler running but lock your workstation. At the scheduled time Macro Scheduler will unlock the workstation, run the macro and then lock the workstation again.

Requirements Summary:
Macro Scheduler Service must be running
Macro Scheduler must remain running

Logging into Windows from Logged Out State

© 2015 MJT Net Ltd

When Windows is logged out Macro Scheduler cannot run. Instead the Macro Scheduler Service will take over scheduling. For this to happen you need to check "Enable AutoLogon when Windows is logged out" under Tools/Options. When Macro Scheduler is closed, (e.g. when Windows is logged out) and this option is enabled, the Macro Scheduler Service will take over scheduling. Macro Scheduler also needs to be in the startup folder, which it is by default after installation. Macro Scheduler will attempt to put itself there if AutoLogon is enabled and it is not already set to run on startup.

Now, when the macro should run, the Macro Scheduler Service will log into Windows with the specified settings, Macro Scheduler will start, run the macro and then log out of Windows again.

Requirements Summary:
Macro Scheduler Service must be running
"Enable AutoLogon when Windows is logged out" must be checked under Tools/Options
Macro Scheduler must be set to run on startup

Note that due to security restraints in current versions of Windows, a service cannot run interactively. Therefore the Macro Scheduler Service can only run macros on a time based schedule. So any window event schedules, or other triggers will not fire when Windows is logged out. This is also why Macro Scheduler needs to be running on startup - the service will log into Windows and then communicate with Macro Scheduler to ask it to run the macro, rather than attempt to fire the macro itself. If the service started the macro itself, or even started Macro Scheduler itself, the macro would not be able to run interactively and would not be able to mimic a user, and respond to and interact with user interface elements etc.

If Macro Scheduler is not set to run on startup, the service would log into Windows but the macro will not run until Macro Scheduler is restarted (and Windows will then be logged out again).

1.15 Logging

To set up a log file for a script select the tab marked 'Logging' on the Macro Properties Window 4.

To enable logging check 'Log Progress of Macro'.

You can have Macro Scheduler purge the log file before each run by ticking the second box.

Enter a file name for the log file or select an existing one by using the browse button. If you like you can use one file for more than one macro.

In Log Level you can specify whether to log just after each script step is executed, just before, or before and after.

1.16 Encryption

The majority of us probably won't ever have to use the encryption facility. However, if you need to use Macro Scheduler to automate a process which involves sending passwords to other applications or to send other sensitive information, then you would want to ensure that only the right people can edit the script and see the secrets.

Macro Scheduler allows you to set a password for a script which must then be used to edit it. The script file itself is scrambled so that if it is edited in any way it wouldn't make any sense.

For encryption options select the Encryption tab of the Macro Properties 4 dialog.

© 2015 MJT Net Ltd

Simply check the "Encrypt This Macro" box and provide a password.

The password must be entered twice to ensure it is entered correctly.

Next time you try to edit the macro you will be asked for the password.

Using the last two options you can specify whether the password should be entered when the macro is run. The first of these two options will ensure that when a macro is started from the main window [2], command line [26], or Macro [204] command, it will only run if the correct password is entered. The second option will further secure the macro for scheduled events, so that a scheduled macro will need the password to be entered before it will start.

To disable encryption on a macro that has been encrypted, edit it, select the Encryption tab and then uncheck "Encrypt This Macro".

1.17 Hot Keys

Each script can be assigned a hot key to allow the script to be launched from a keyboard shortcut.

To assign a hot key select the tab marked 'HotKey' from the script settings window and select the appropriate keys from the drop down lists.

Under Scope select whether this macro should be active all the time, or if it should only be active if the macro's group is the active group. If the second option is selected it is possible to assign the same hot key to another macro in a different group.

Hot keys are system-wide. In other words Macro Scheduler does not have to be active or visible for them to work and they can be triggered even when another application has the focus.

1.18 Groups

Macros can be organised into groups. Groups can be created using the Groups menu [3], Toolbar [4] or context sensitive menu in the groups pane of the Main Window [2]. Macro groups can be assigned a physical path in Group Properties [33]. You can then drag macros from group to group.

You can also set a macro's group in Macro Properties under the Group tab.

If 'Show In System Tray Quick Launch Group' is checked in Macro Properties the macro will appear in the pop-up menu that appears when 'Quick Launch' is selected from Macro Scheduler's system tray icon. By default new macros are added to the Quick Launch group. Remove a macro from the Quick Launch group by unselecting this option.

Groups can be re-ordered and re-structured by dragging and dropping them between nodes in the group tree. To move a group to become a sub-group of another drag it on to that group. To re-order groups at the same level drag and drop a group to the left of the one you want it to appear before.

Linked Groups

It is possible to create "Linked Groups". Linked Groups are designed for collaboration and sharing of macros across a network.

To create a linked group check the "Create Linked Group" checkbox in Group Properties when creating a new group and choosing an existing path containing macro files. Any macro files that are

© 2015 MJT Net Ltd

already in the specified path will be displayed in the group. If any macros are added to the folder at any time they will appear in the group.

You cannot delete linked macros or move linked macros to other groups (because they belong to someone else and would just appear again anyway).

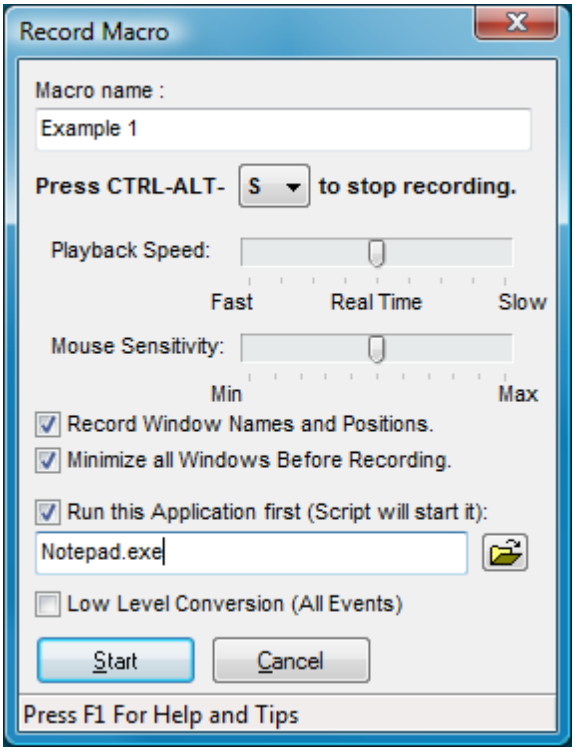
1.19 Stats

Displays runtime statistics for the macro such as number of executions, last run date/time, last elapsed run time, number of lines executed and average execution time.

Also provides a basic savings calculator for determining time and cost savings of the automated versus manual process.

1.20 Recording Macros

To record a macro select Run/Record, or press the 'Record' Toolbar 4 button (looks like a video camera) button on the Main Window 2. You will be prompted for a name for the macro and recording will commence when Start is pressed.



By default CTRL-ALT-S will terminate the recording. You can select an alternative key from the drop down list box if required. SHIFT-ESC (or the shortcut chosen in Options 37 to stop running scripts) will also stop the recording in the same way that it stops playback.

Carry out the tasks you want to be captured and finally press CTRL-ALT-S (or chosen key) or click on 'Stop' to end the recording.

The new macro will appear in the macro list and can be executed by clicking the 'Run Now' button.

You can modify the playback speed using the slider. This applies a modification to the Wait times between individual events. In many cases, attempting to speed the macro up by reducing these times will actually make the macro less reliable. It is advisable to leave the speed at real time.

The mouse sensitivity setting is used to ignore small mouse movements. Modify the degree of sensitivity by adjusting the slider. The further towards Max, the more sensitive the recorder is and the smaller the movements that are ignored. By ignoring small mouse movements the macro recorder can refine the script where small, unnecessary mouse movements appear between other events which can be combined into one script command. If these small movements are not ignored then the individual events need to be handled separately, so lengthening the script. If in doubt leave it in the default, middle setting.

With the 'Record Window Names and Positions' option selected the macro recorder will notice when

© 2015 MJT Net Ltd

your actions cause windows to appear and insert code to make sure the macro waits for those windows to appear before continuing. It will also add code to ensure those windows appear in the same positions and with the same dimensions as they appeared when you recorded the macro. This ensures the macro will work correctly each time with your mouse events landing in the right places.

When recording a macro for a specific application, select the application to run. Macro Scheduler will run the application and remember it's name and position and add the necessary commands to the top of the recorded macro to ensure the application is executed, sized and positioned in the same place each time, so ensuring the reliability of the recorded macro.

The last setting - 'Low Level Conversion' - will avoid parsing to result in a script of lower level commands. E.g. an LDown and LUp will be left alone and not merged into an LClick command.

Recommendations:

- To make a reliable macro we recommend using the "Run this application first" option. The macro recorder will then start that application and the generated script will also ensure it is started when the script is run. This is a far more reliable way to start an application than by double clicking on a desktop shortcut icon, for example. Don't record pointless mouse clicks to start an application when the application can be started directly. Let the macro recorder start the application and have it generate a script that runs that application specifically before the rest of the macro proceeds.
- Use the "Record Windows Names and Positions" option to make the generated script wait for target windows to appear; and size and position them the same way each time.
- Minimize all windows before recording your macro so that you are starting against a "clean" desktop and to avoid recording window names that are present now but may not be during playback.
- Learn how to write scripts manually. Written macros can be made more efficient, more reliable and more aware. Read the scripting tutorial 41.

1.21 Playing Scripts And Macros

To play a macro or script without scheduling it use the 'Run Now' Toolbar 4 button on the Main Window 2. You can also press CTRL-R, or choose Run/Run Now from the Main Menu 3. Another way to start a macro is to right click on it and choose 'Run Now' from the pop-up menu.

After starting a macro the stop button on the tool bar will become enabled. You can press this to stop the macro at any time. CTRL-B or Run/Stop will also stop a macro.

To stop macros no matter what program or window is active, press SHIFT-ESC. You can change this shortcut in Options 37.

The menu that pops up from the icon in the task bar also has an option called 'Stop'. This works like the stop button and allows you to cancel the execution of a script. This option is available even when a script is executed automatically by the scheduler.

To pause all running scripts during execution press the Pause key. Pressing the Pause key again will resume paused scripts. You can specify an alternative to the Pause key in Options 37.

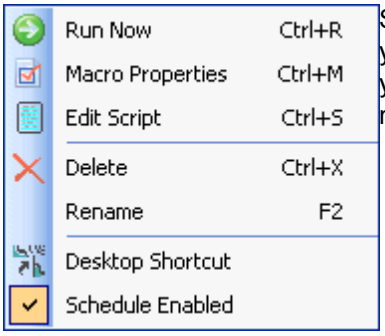
You can run other macros within scripts by using the Macro 204 command.

Macros can also be assigned to desktop shortcuts or run from the command line. See Creating Desktop Shortcuts 22 and Command Line Option 26.

© 2015 MJT Net Ltd

1.22 Creating Desktop Shortcuts

You can tell Macro Scheduler to create a shortcut for a macro by selecting the appropriate macro from the main window and then clicking the right mouse button to display a pop up menu. Alternatively you can select Tools/Desktop Shortcut from the Main Menu 3, or press the Desktop Shortcut Toolbar 4 button.



Select the second from last option and a shortcut will be placed on your desktop. To run the macro you then only need to double click your desktop icon. Once it is on your desktop you can, if you prefer, move it elsewhere in the usual way using explorer etc.

1.23 Using the Debugger

In the Editor there is a menu called 'Debug' with the following options

Step (F8)
This highlights the currently selected line, and then subsequently executes that line and moves to the next line in execution flow.

Step Over (Shift-F8)
This option is enabled if the current line contains a GoSub command or an If statement which contains a call to a subroutine. Step Over executes the subroutine in whole and moves the debug cursor to the next line.

Stop Debug (CTRL-F2)
Stops debugging and resets everything.

Trace
This displays a small dialog where you can set an interval. Then the debugger auto-steps at the specified frequency. You can set the interval as low as 0 seconds although if set to 0 seconds the debugger actually runs each line with a delay of 0.2 seconds between them in order to give you a chance to see what is happening and interrupt the script if needed. It is recommended to disable Refocus Windows (see below) when using Trace so that focus does not switch back to the debugger between each line.

Run
Runs the script without stepping, from the currently selected line. The script will run to the end or until the next breakpoint, from where the script can be stepped or run.

Insert Breakpoint
Inserts a breakpoint after the selected line. Simply inserts **BREAKPOINT** on the next line. If running the script using 'Run' from the Debug menu, execution will pause when this Breakpoint line is reached and the script can then be stepped from this point. As of version 10 multiple breakpoints can be set. At any time pressing Run will run the script to the next breakpoint from where the script

© 2015 MJT Net Ltd

can again be stepped or run.

Variable Breakpoints
Allows you to set one or more variable values on which script execution to break. Specify name=value pairs. E.g. to make the debugger pause when X=5 enter X=5 in the list. Then, when you run the script execution will pause when X=5 no matter where that happens. The script can then be stepped or continued from this point.

Run From Top
With this item enabled the script will always be run from the first line when it is started. With this option unchecked the script will be started from the line containing the cursor. This can be useful when debugging scripts and you only want to run a section of code.

Refocus Windows (check on or off)
Most of the time you want this checked on. Due to the nature of Macro Scheduler, and the fact that it is commonly used to automate other windows, most scripts will focus other apps and chop and change focus a lot. 'Refocus Windows' ensures that if a command causes focus to shift, focus will be set back to that window before executing each subsequent line so that (hopefully) that line will affect the correct app. Focus is returned to the editor after each line so that you can see what you're doing. Sometimes it is useful to turn this off ... if you're not doing any GUI scripting, or even at some point during a GUI script where you might have a loop, say, calculating some value but not actually requiring focus of anything. So you can switch it on and off during debug as required.

Show Watch List
Displays a list to the left hand side of the editor containing all current script variables and their values - updates as they change. This is the most useful debugging device as you can see the values of variables as they are set. By default the watch list shows the most recently modified variable at the top of the list. Or you can sort the watch list by right clicking and selecting "Sort". Right clicking on an item allows you to copy the line or just the variable value to the clipboard. You can also search for a variable or value by typing a value into the search bar at the top of the watch list.

Open Watch List on Debug
With this option enabled the Watch List will always open when the script is started.

Lock Watch List Position
As the script executes new variables are added to the watch list as they are created. Depending on how you have the Watch List sorted this will mean the position of variables within the watch list will change. E.g. new variables appearing at the top. If you want to keep your eye on a particular variable and not have it obscured when a new variable is added, enable Lock Watch List Position to prevent the Watch List from scrolling and keep the line you are interested in visible.

Debug Line Color
Allows you to change the color that is used to highlight the active script line.

Notes
Remember that stepping through a script will slow it down, since you will be pausing at each line. Since many scripts that automate windows applications need to be time-sensitive - to make sure things don't happen before apps are ready etc - the process of debugging may give the impression that the script is fine, when it actually needs delays and waits to be built in!

Using the Watch List

You will notice that variables in the watch list are preceded with a number, starting at 0. Unless you use LOCALVARS 266 you will never see any other value. With LOCALVARS enabled this number changes depending on the scope level. E.g. a global variable called MyVar will be at level 0 while a local variable with the same name will be 1 or more. The higher number relates to nested level of the subroutine.

© 2015 MJT Net Ltd

Arrays will show (Array) after them. Double click on array variables to see their element values.

You can modify a variable on the fly during debug before stepping or after reaching a breakpoint. Right click on a variable and select Modify... to change it's value.

You can also copy a line or value or sort the watch list using the right click menu.

Other Debug Methods
Other things you can do to help debug your script include:

- Use LogFiles - See Logging 18 options
- Use MessageModal 170 command to display diagnostic information on screen
- Use _DUMP_VARS variable 269 to dump all variable values to the log file
- Use the Variable Explorer 272 (Tools/Variable/Code Explorer or CTRL-ALT-V) to view all user variables and find out where they are created/modified. This also lists all subroutines and dialogs.
- Use the STEP_DELAY 269 variable to slow the script down.

For an introduction on using the debugger watch this video:
http://www.mjtnet.com/demos/using_macro_schedulers_debugger.html

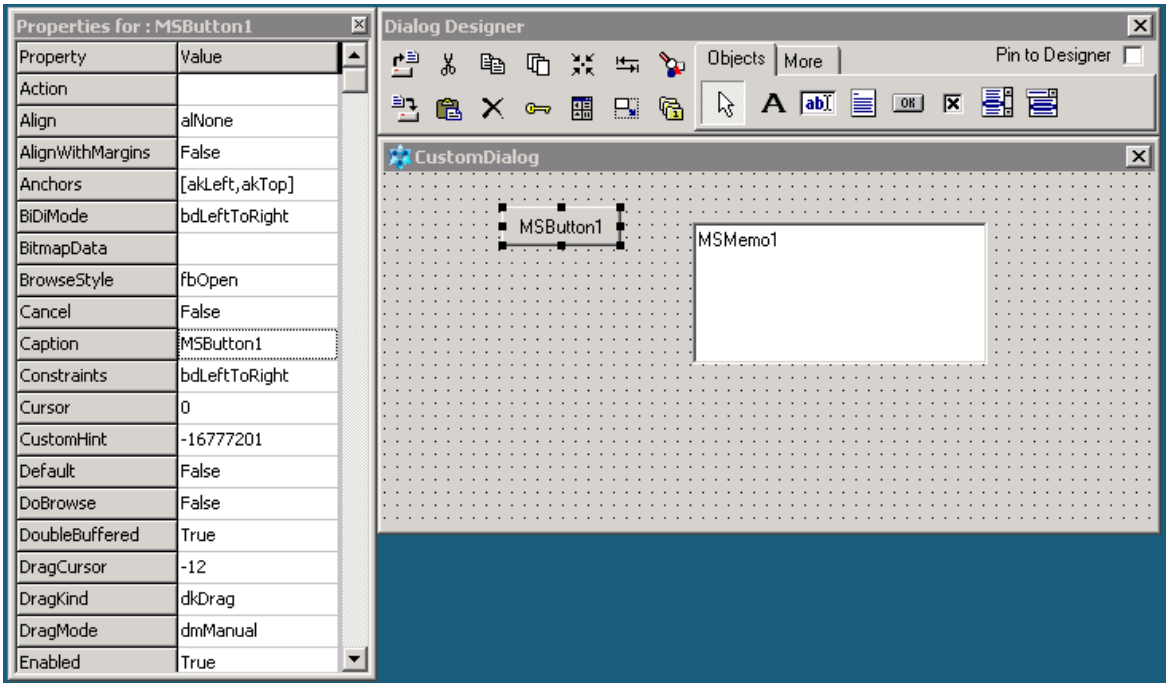
1.24 Dialog Designer

This is accessed from the Tools menu in the Editor 4. This is an easy way to create MacroScript dialogs.

If a Dialog 102 block is highlighted when the Dialog Designer is selected, that dialog will be displayed in edit mode.
Otherwise a new dialog is displayed.

You can also edit a dialog by right clicking on the first line of the Dialog block and selecting the edit option from the pop-up menu.

© 2015 MJT Net Ltd



Select objects from the Objects bar to add to the dialog. Objects can be moved around and resized.

Toolbar buttons exist to copy, cut, paste, lock, resize, and move the objects. You can also modify the tab order.

When you click on an object you will see it's properties in the properties inspector to the left. Right click on an object to see more options such as font settings and also to view a list of event names available for the object for
use with the AddDialogHandler 106 function to create an event handler (e.g. to react to a button being clicked etc). Some objects may have different options available. E.g. for page controls use the right click menu to add pages and navigate through pages.

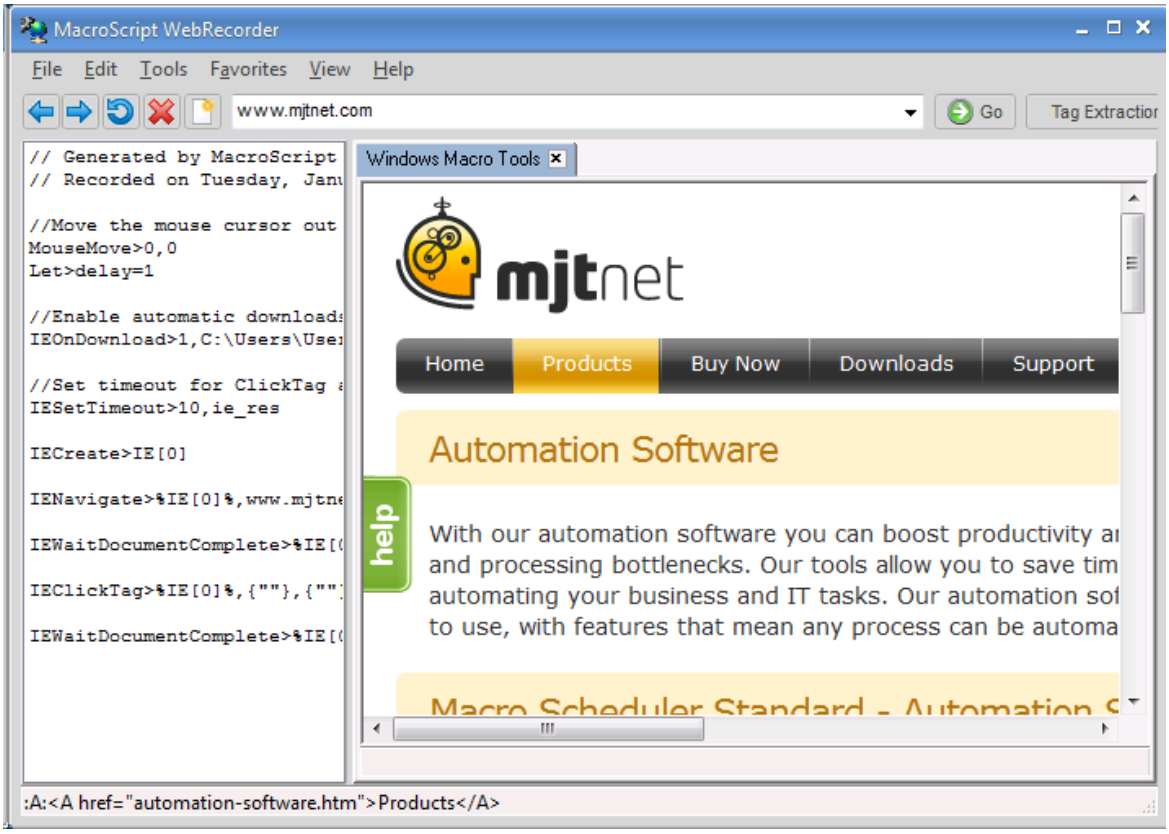
The bottom left toolbar button saves the dialog to the editor. If the dialog has not already been saved (does not already exist in the editor) it will save the dialog block code to the current cursor position. A message will pop up first giving you the chance to cancel and reposition the cursor if needed. We recommend storing dialog blocks at the top of script before your main logic. If the dialog block already exists in the editor it will be updated.

1.25 WebRecorder

WebRecorder assists in the creation of scripts that control websites and web pages. It generates code that uses some of the WebRecorder functions.

You can start WebRecorder either via the shortcut in the Macro Scheduler program group, or from within Macro Scheduler from the toolbar option in the Editor 4.

© 2015 MJT Net Ltd



Simply use WebRecorder as you would a web browser. As you navigate around the web page you will see code produced in the left hand code pane.

Once you have finished you can either save the code to a script, copy and paste it, or - if you opened WebRecorder from within the script editor simply closing it will cause the new code to be inserted into your script.

You can also run the code from within WebRecorder.

The Tag Extractor assists in creating code that pulls data out of the web page. Start the Tag Extractor using the Tag Extractor button in the tool bar. Then point at an element and you will see information about that element displayed. Click Insert to insert code into the script that will extract that data at runtime.

WebRecorder only uses a small subset of the functions available and has to make certain assumptions which may not always be correct. You need to be prepared to edit the code that is produced and using the other commands you have more control over how elements are identified. See WebRecorder Functions 229.

1.26 Command Line Options

It is possible to run macros from the command line using the following syntax:

msched macroname

e.g. to run the Defragment Disk example script you would type:

msched Defragment Disk

© 2015 MJT Net Ltd

This is useful for creating shortcuts and running Macro Scheduler scripts from other programs or from macros created in other applications such as Word or Excel.

However, if you want to create a shortcut, you can get Macro Scheduler do it for you. See Creating Desktop Shortcuts 221.

Parameters

When running a Macro from the command line in this way, you can also pass parameter values into the script :

msched Example Script /filename=testfile.txt /path=c:\outpath

The above example runs a script called 'Example Script' and passes two variable values in filename and path. These variables can be used in the usual way within the script, e.g. :

```
Change Directory>path
ifFileExists>filename,ok
Goto>end
Label>ok
Message>File Exists !!
Label>end
```

It is also possible to run any file by passing the full path and filename to Macro Scheduler like this :

msched c:\scripts\my script.scp

In this instance, the script 'my script' does not have to exist in the Macro Scheduler macro list. If you wish you can associate .scp files with Macro Scheduler so that Macro Scheduler scripts can be run by double clicking on them in explorer.

The same method can be used for compiled scripts:

compiledmacro.exe /fullname="Donald Duck" /filename=testfile.txt

Note that quotes can be used to surround values that contain spaces.

Log File

You can specify a log file for the macro by passing the LOGFILE variable:

msched.exe c:\scripts\my script.scp /LOGFILE=c:\mylogfile.log

As this is passed as a script variable it will also be available to the macro as a regular variable.

It is not necessary to specify a log file if running a macro using only the macro name for a macro that already has logging enabled. In this case the log file settings for that macro will be used. However, if the LOGFILE parameter is passed it will override the macro's existing logging settings. The LOGFILE parameter can also be used with compiled scripts.

Normally each time a line is added to the log file the log file is opened, the line is written and the file is then closed again. This ensures reliability and prevents possible issues with files being left open or locked if a problem should occur mid way through the script. However in some very demanding production scenarios (e.g. thousands of multiple running scripts logging to the same file) this can introduce a performance penalty. In this situation you may want to keep the log file open during script execution. This can be done by adding the KEEPLOGOPEN switch to the command line.

© 2015 MJT Net Ltd

Script URLs

The script file can also be an HTTP URL, so that you can run macros from the web:

msched.exe http://www.mjtnet.com/scripts/sample.scp

If you need to specify a username and password to access the script do so as follows:

msched.exe http://username:password@www.mjtnet.com/scripts/sample.scp

Opening the Editor

To run Macro Scheduler in macro editor mode, use the -EDITOR switch:

msched.exe -EDITOR

To open a script file directly in the Macro Scheduler editor, add the filename to the command line:

msched.exe -EDITOR c:\myscripts\somescript.scp

Disabling System Tray Support

To switch off system tray support, so that the Macro Scheduler icon does not show in the System Tray add the parameter -NOSYSTRAY or /NOSYSTRAY at the end of the command line :

msched -NOSYSTRAY

Hiding Macro Scheduler

You can hide the Macro Scheduler window with the -HIDE parameter:

msched -HIDE

To hide Macro Scheduler completely use this in conjunction with the -NOSYSTRAY parameter:

msched -HIDE -NOSYSTRAY

-HIDE and -NOSYSTRAY can also be used with compiled macros.

1.27 Help and Resources

By far the best place to get support is at the Macro Scheduler forum:

http://www.mjtnet.com/usergroup/

Also, keep an eye on Marcus' blog:
http://www.mjtnet.com/blog/

On the blog you will find articles on how to work with databases, how to use image recognition, working with Microsoft Excel, using the debugger, how to get started with automation and lots more. The blog is updated regularly, so keep an eye on it or subscribe to the RSS feed.

For more support resources or to contact us please visit:

© 2015 MJT Net Ltd

http://www.mjtnet.com/support.htm

Please also send bug reports, comments and suggestions via the above page.

For hints and tips have a look at the Scripts & Tips page at:

http://www.mjtnet.com/scripts.hts

Finally, keep an eye on the MJT Net Ltd web site for new product announcements and information:

http://www.mjtnet.com

The online version of this help file also has a comments feature where comments, questions, tips and examples can be posted. Each topic in this help file has a link to the online version.

© 2015 MJT Net Ltd

Menu Commands

Part II

2 Menu Commands

2.1 New Macro

Displays the Macro Properties window for creating a new macro.

2.2 New Macro From Template

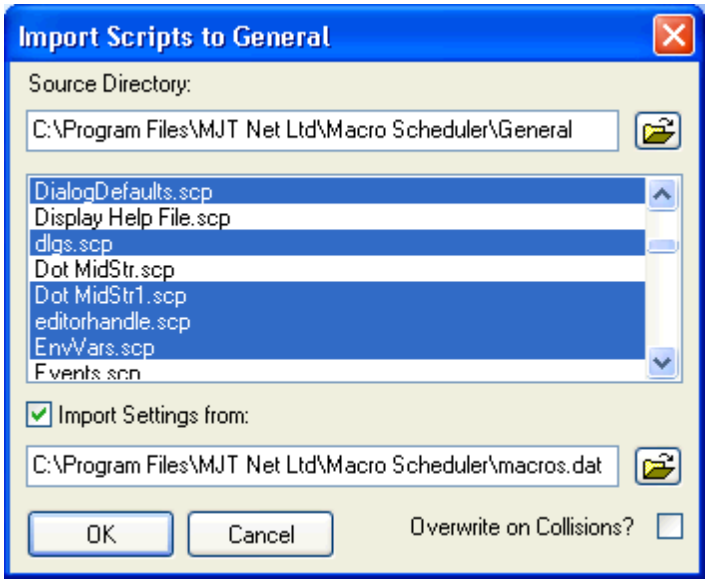
Allows you to create a new macro using an existing template/.scp file. By default the open dialog displays .scp files in the Templates subfolder of your Macro Scheduler program folder. A number of templates are provided with the software. You can create your own .scp files as templates, or use New Macro From Template to make a new macro from any existing .scp file.

2.3 Edit Macro

Displays the Macro Properties window for the currently selected macro.

2.4 Import Macros

Allows macros from an external directory or another Macro Scheduler installation to be imported into the currently selected group.



Locate the source directory and highlight the files to be imported.

The files will be copied into the current group directory and will appear in the macro list. To import script settings, such as schedules, hot keys and logging details, select 'Import Settings from' and locate the macros.dat file that contains the data.

During the import process if a macro with the same name already exists in Macro Scheduler the new name will be suffixed with a numeric value to make a name that is unique.

'Overwrite on Collision' relates to physical files in the directory, not macro names. If a target file

© 2015 MJT Net Ltd

already exists it will be overwritten only if 'Overwrite on Collision' is checked.

Please note: Import Macros can not successfully copy encrypted macros. If you wish to import encrypted macros please first decrypt them.

You can also import macros simply by dragging .scp files onto the main Macro Scheduler window.

2.5 Hide

Hides the main form (minimizes to the system tray). Not available in NT3.5x

2.6 Exit

Closes Macro Scheduler.

2.7 Delete

After prompting for confirmation, deletes the selected macro.

2.8 Remove from Group

Removes the selected macro from the group. This differs from Delete 32 in that this option does not delete the macro's script file from the disk. The macro is simply removed from the list of macros in Macro Scheduler. It's script file still exists on the drive.

2.9 Rename

Displays the rename dialog to allow the selected macro to be renamed.

2.10 Select All

Selects all macros in the macro list.

2.11 New Group

Select this to create a new macro group. The new group is created beneath the currently selected group. Therefore, to make a new top level group, select 'All Macros' first.

On selecting this option the Group Properties 33 dialog is displayed where the new group name and a macro path are selected.

A macro path is simply a folder on a drive or network drive which is used to store the physical files associated with macros that you place in the associated group. By default the main Macro Scheduler directory is used.

Linked Groups

© 2015 MJT Net Ltd

It is possible to create "Linked Groups". Linked Groups are designed for collaboration and sharing of macros across a network.

To create a linked group check the "Create Linked Group" checkbox in Group Properties when creating a new group and choosing an existing path containing macro files. Any macro files that are already in the specified path will be displayed in the group. If any macros are added to the folder at any time they will appear in the group.

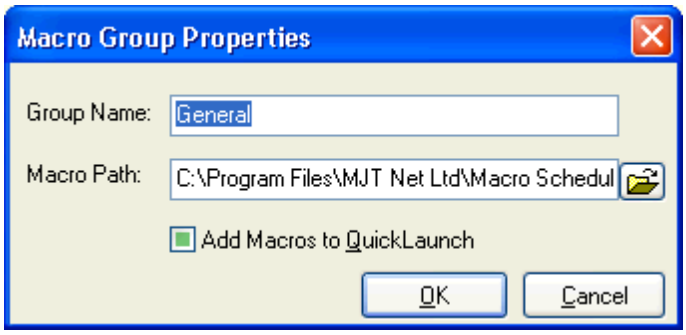
You cannot delete linked macros or move linked macros to other groups (because they belong to someone else and would just appear again anyway).

2.12 Delete Group

Deletes the selected group. A group containing macros cannot be deleted. If an attempt is made to delete a group that has macros a message box will be displayed to this effect. Delete or move any macros before deleting the group.

2.13 Group Properties

Displays the group properties for the currently selected group. The name of the group and/or the macro path can be changed.



You can add or remove all the group's macros to and from the Quick Launch system tray menu in one go by checking or unchecking 'Add Macros to QuickLaunch'.

2.14 Sort

Presents a further menu containing the names of the macro list columns. Selecting one sorts the macro list by that column.

You can also sort the macro list by clicking on a column header.

2.15 Font

Displays the standard font dialog to allow the macro list font to be set.

© 2015 MJT Net Ltd

2.16 Grid Lines

Toggles the grid lines in the macro list on and off.

2.17 List

Switches the macro list into list view, showing just the icon and macro name.

2.18 Details

Switches the macro list into classic details mode, showing all information in columns.

2.19 Large Buttons

Toggles the toolbar buttons between regular small image only buttons, and larger buttons with text as well as images.

2.20 Toolbar

Shows or Hides the toolbar.

2.21 Search Bar

Shows or hides the Search Bar. Use the Search Bar to search for macros. The Search Bar searches inside the script file as well as the name of the macro.

2.22 Create Exe

If you have the Professional edition installed this option will create a standalone executable version of the highlighted script. Some functions require runtime library files which must be in the same folder as your .exe. See "Runtime Libraries" below for details.

Output file:
The compiler will prompt for an output exe file. By default the exe will have the same name and path as the script being compiled. You can browse for a different location using the browse button.

Icon:
As an option a custom icon can be embedded into the compiled executable by selecting an icon (.ico) file. This is not supported in Windows 95/98. To compile a custom icon into the executable use Windows NT, 2000, XP or above.

Options:

© 2015 MJT Net Ltd

Create Console App	Creates an EXE which runs as a console app and can write to STDOUT
Compile Includes	Scripts referenced by Include> statements will be embedded and compiled into the Exe. Will not work where Include> references scripts via variables other than SCRIPT_DIR.
Disable Logging	Prevents the Exe from being logged (adds /LOGFILE=\dev\null to Include Parameters)
No System Tray Icon	No system tray icon will be created (adds /NOSYSTRAY to Include Parameters)
Run Hidden	Task bar icon will not be shown (adds /HIDE to Include Parameters)
Disable SHIFT +ESC Stop Key	Prevents SHIFT+ESC being used to stop the Exe (adds /NOSTOPKEY to Include Parameters)
Copy Runtime DLLS	If a command that needs runtime DLLs exists in the script those DLLs will be copied from the program folder to the .exe location
Copy BMP Subfolder	If the script uses image recognition its BMP_DIR (used to store needle images) will be copied to the destination

Include Parameters:
Optionally you can also specify a command line to include in the compiled executable. This means command line parameters can be "hard-coded" into the executable so that they don't have to be specified on the command line when the executable is run. Any of the command line options that can be passed into scripts can be included, including parameters to be passed into scripts. E.g. /HIDE, /NOSYSTRAY, /LOGFILE= and script parameters can be hard coded into the executable.
See Command Line Options 26.

The title of the compiled executable can be modified by setting the APP_TITLE variable in the script or pass it in Include Parameters. The default is "Macro Scheduler". E.g.: Let>APP_TITLE=My Program

The executable file created by the compiler can be run on any machine without having to have Macro Scheduler installed, and it will run the script.

Compiling from the command line:
To compile on the command line run msrt.exe as follows:

msrt.exe -COMPILE source.scp target.exe [-QUIET] [-DEL] [-OPTS:options] [-ICON:iconfile]

Optional switches are shown in square brackets and are as follows:

-QUIET will prevent compiler error/success messages being displayed
-DEL deletes the source script file
-OPTS: then a command line to compile into the executable, e.g.: -OPTS:-NOSYSTRAY /parm1=fred
-ICON: then a .ico file to change the default program icon

To compile a console app use msrt_console.exe instead of msrt.exe.

Runtime Libraries

If your script contains any of the text capture functions (GetTextAtPos, GetTextInRect, GetWindowsTextEx, GetTextPos, WaitScreenText) you will need to supply the following files with your .exe:

GetWord.dll
GetWordNT.dll
GetWord_x64.dll
GetWordNT_x64.dll
GetWord_x64.exe

© 2015 MJT Net Ltd

License.dat

These files can be found in the Macro Scheduler program folder. Copy the above files to the same location as your .exe, or use the compiler option.

Stopping Compiled Macros

Unless disabled (see compiler options above) SHIFT-ESC will normally stop a running .exe. However, since this uses the Windows hotkey system and only one application can "own" a hotkey at a time, if some other application has already registered this hotkey it will fail to stop the .exe. Also note that Macro Scheduler itself uses this stop key sequence by default, so if you already have Macro Scheduler running and have not changed the stop key sequence under Tools/Options then it will fail to stop the .exe as Macro Scheduler will get it first. It is unlikely that your users will also be running Macro Scheduler.

Various other options exist for creating your own custom exit methods. E.g. you could use an OnEvent KEY_DOWN handler in your macro to create your own stop key sequence specific to your macro. You could even give your users a choice of keystroke combination. Another example would be a small custom dialog with a stop button.

2.23 View Log File

If the selected macro has a log file assigned to it, this option will display that log file in Notepad.

If you wish to use a different viewer application you can add the following key to the registry:

HKEY_CURRENT_USER\Software\MJTNET\MSched - LogViewer

Set the LogViewer value to the path of the log viewer application/editor.

2.24 Desktop Shortcut

Creates a desktop shortcut for the selected macro. See Desktop Shortcuts 221.

2.25 View System Windows

Displays a tree of all windows currently open in the system. The tree represents a hierarchy showing the handle, class name and title text for each window and their child windows.

You can search through the tree by entering some text in the Find box and pressing Enter or clicking on 'Find'. Press Find again to locate the next match.

Refresh the list by clicking the 'Refresh List' button.

By right-clicking on an entry you can:

- Copy the object's caption - this is the text of the object
- Copy the entire text shown in the list (handle - class name and object text)
- Show the selected window
- Hide the selected windlw

© 2015 MJT Net Ltd

- Close the selected window - use this with caution.

2.26 Options

Displays the options dialog. Options are:

Stop Running Scripts With
By default SHIFT-ESCAPE will stop running scripts. You can change the shortcut used to stop running scripts here by entering the key sequence you prefer.

Pause Running Script With
By default the PAUSE key will pause/resume running scripts. You can change this key sequence here.

Start Minimized
To make Macro Scheduler startup minimized, check the box - 'Start Minimized'.

Close Button Minimizes
If you want the close button to minimize Macro Scheduler instead of closing it, check the box marked 'Close Button Minimizes'

Allow Multiple Instances
By default it is possible to have more than once instance of Macro Scheduler running at the same time. However, if you need to ensure that only one instance can run at a time, uncheck the third option 'Allow Multiple Instances'.

Escape Key Cancels Macro Properties Dialog
If you want to be able to cancel changes when using the Macro Properties dialog just by pressing the Escape key then enable this option. For safety this is disabled by default.

Warn about trailing spaces when saving macros
With this option checked Macro Scheduler will warn you when saving a macro of any lines that have trailing invisible characters. If no lines have trailing invisible characters no warning will be made. In any case you can switch this check off altogether by unselecting this item.

Enable AutoLogon when Windows is logged out.
This option requires Windows Vista, Windows Seven, Windows 2008 Server or above.
If this option is checked the scheduler service will continue to schedule macros when Windows is logged out and, if AutoLogon 17 is enabled for a scheduled macro, will automatically log into Windows, run the macro and log out again.

Show Running Indicator when Hidden
When a macro is started while Macro Scheduler is hidden (i.e. via the scheduler or from a hotkey) a balloon tip will appear by the system tray icon indicating which macro was started.

Attempt to Deactivate Screen Saver on Run
By default when a macro is started Macro Scheduler will try to deactivate any running screen saver, as screen savers can prevent windows from being focused and receiving keystrokes etc. Uncheck this option if you would prefer not to deactivate the screen saver, but bear in mind that any scripts that need to interact with windows may not work correctly if a screen saver is running.

Insert directive comments in new macros
By default when you create a new macro some useful comments are added to the top of the script regarding some directives that can alter the way the script is parsed. These may be useful for newcomers but if you would prefer them not to be added then disable this option.

© 2015 MJT Net Ltd

Disable Scheduler while editing/debugging scripts
By default schedules continue to occur while the editor/macro properties is open and macros are being edited or debugged. This can be an interference. Check this option to disable the scheduler while you are editing a macro. Repeating macros will not be triggered but will have their schedules pushed forward to the next interval so that they are not missed after you have stopped editing.

Change Macro/GroupSettings Path (Advanced)
Clicking this button reveals a dialog that allows you to change the default script path and macro/group settings location (where schedules and other macro properties and group settings are stored). Use this either to start a fresh profile in a new folder, move your existing settings files to a new location or link to an existing profile such as one from a previous version. Note that this does not move your actual script files because the location of these is determined by the path given to their group. This option simply changes the default location for new groups and the location where group settings and macro settings (schedules, hotkeys and so on) are stored. To move script files to a new location modify their path under Group Properties 33.

2.27 Run Now

Runs the selected macro. See Playing Scripts & Macros 21.

2.28 Record

Displays the record dialog. See Recording Macros 20.

2.29 Stop

Stops running macros. SHIFT-ESC (or the alternative shortcut chosen in options 37) will also stop macros without Macro Scheduler having to be active. There is also a break option on the system tray menu.

2.30 Contents

Displays the help file contents.

2.31 Check for Update

Use this option to check if an update to Macro Scheduler exists. This option starts your Internet browser and gives you information about updates.

2.32 Support Forums

Starts your internet browser and takes you direct to the Macro Scheduler support forums.

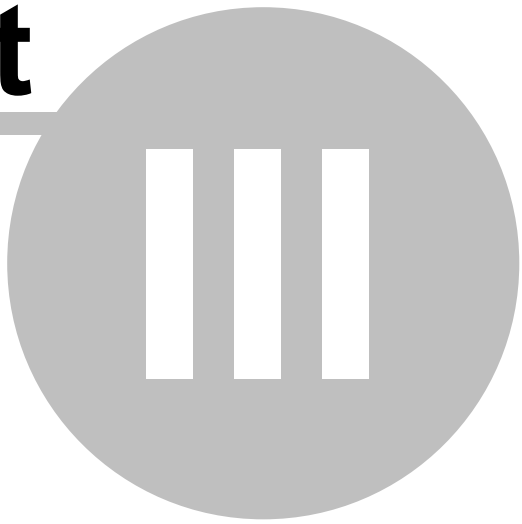
© 2015 MJT Net Ltd

2.33 About

Displays the About dialog box. This is useful to determine which version of Macro Scheduler you are using, who it is registered to and for a quick link to our web site.

© 2015 MJT Net Ltd

Part



3 Scripting Windows For Beginners

3.1 Introduction

Macro Scheduler provides a simple but powerful language for automating Windows applications. This language is similar in many respects to the BASIC programming language. As well as the usual programming constructs it provides specialised commands for automating Windows applications at the user interface level, such as sending keystrokes, mouse events and activating windows.

This guide works through a simple automation example. The scenario is a rather pointless "Hello World" type of situation, but it will work unmodified on any Windows PC and aims to demonstrate key methods used in almost every other automation scenario, and to introduce a number of important script elements.

You can watch a video version of this tutorial here:
http://www.mjtnet.com/demos/tutorial.html

3.2 Requirements

This document assumes you are running Macro Scheduler 6.0 or above on a standard PC running Windows 2000 or above with a default printer enabled. You should be familiar with the Windows operating system. Some basic programming experience is helpful but not essential.

3.3 The Problem

We need to build a script, which opens Notepad, writes a few lines of text, saves the document with a given name and prints it to the default printer before finally closing Notepad. We will look at some of the issues that could crop up – for example what to do if the file already exists and different methods of dealing with this.

3.4 Using Macro Scheduler

This guide is concerned with introducing script development and does not concern itself too much with using the Macro Scheduler interface. Refer to Using Macro Scheduler 2 for a tour around the Macro Scheduler interface.

However, we will assume that you are using the Macro Scheduler editor to build the example script here. Do the following to create the macro and start editing:

1. Double click on the Macro Scheduler icon in your system tray.
2. Select "New Macro" from the File menu.

Double click on the gear icon to view Macro Scheduler's control panel.

Select File/New Macro from the main menu or click the New toolbar button. This will open an empty new macro and put you directly in the Script Editor 4.

While in the editor pressing F1 will display the help topic for the command on the current line or the highlighted command. There is also access to the useful command locator, which makes finding the function or command you need easy. Help is also available from the command locator. Just press

© 2015 MJT Net Ltd

F1 while a command is selected.

3.5 First Steps

Usually, we'll already be familiar with the process that needs to be automated. This is important as Macro Scheduler works by simulating user input, and since we have to tell it what keys to press, and which apps to activate, then we need to know what these are in the first place. It is also advisable to try to avoid mouse events, as these are subject to window placement, which can vary. Although there are ways round this, such as running apps maximised, and using Macro Scheduler's relative mouse move commands, it is usually possible to use the alternative keyboard shortcuts, and in most cases this improves the reliability and efficiency of scripts.

Therefore it is always helpful to run through the process manually, making a note of the key presses needed to perform each action. Note down titles of the windows that appear, how long certain actions take and if there is anything that signifies that an action has completed. The list of key presses and windows is the basis of our script.

It is probably best to break the process up into manageable chunks. In our example the first thing we need to do is to run Notepad and wait for it to be ready. Then send the lines of text. We can safely develop this portion of the script before we begin to consider the next sequence.

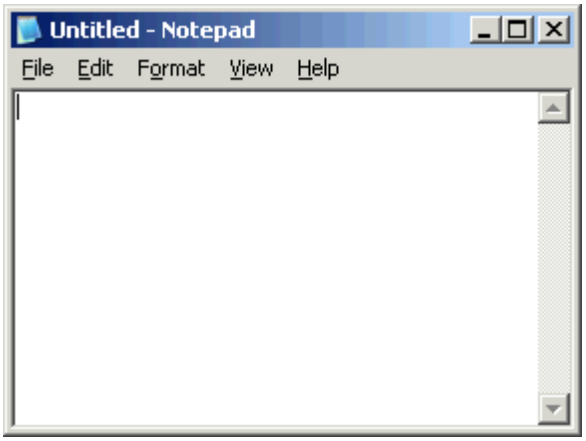
3.6 Building The Script

Let's make a list of what our script needs to do:

Run Notepad
Wait for Notepad to be ready for input
Send the lines of text
Save the document
Print the document
Close Notepad

Steps 4 and 5 will probably need to be further broken down – they will involve some key sequences and dialogs. Let's run through the process manually.

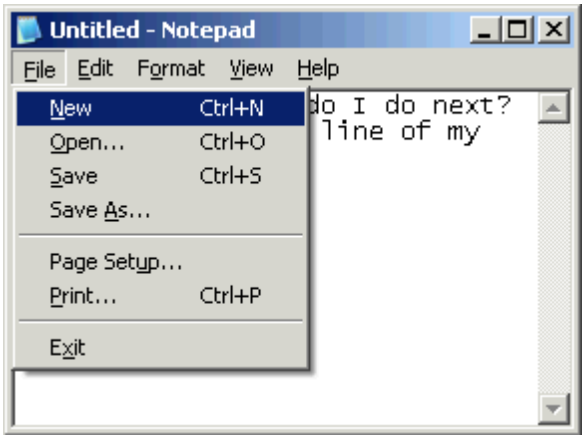
This is what Notepad looks like when we run it. The first thing to note is the Window title:



"Untitled – Notepad"

Notice that focus is already in the editor portion (i.e. the editor is already active), so to send our lines of text we can just go ahead and start typing.

© 2015 MJT Net Ltd



So I've written some text and now I want to save the document. Let's see what the key sequence is for this.

ALT-F opens the File menu.
ALT-A will perform the Save As.. action.

Therefore ALT-FA will do this in one go.

Remember: ALT-FA

The ALT keystrokes that issue a command are underlined on Windows. So, when Notepad is opened, the first item is marked File – this tells us that if we press 'ALT' and 'F', the File menu will appear.

So, back to our example. When we press ALT-FA we get the "Save As" dialog box. Note that the window title is "Save As" and the focus is already in the filename box – this will be helpful.

So when we do this manually, we are subconsciously waiting for the Save As dialog box to appear and then we can type the filename. Normally most of us would probably use our mouse to locate the folder we want to save the document in and then provide a filename. However, we can, of course, type the full path and filename directly into the filename box and just press Enter. Assuming we were able to get this right first time it would work. Well, of course, our script will manage this happily because we're going to define the filename up front.

Notice that the Save button on the Save As dialog is the default object – this means that just Pressing Enter will save the file.

So we have determined the actions required for the first part of our process:

- Run Notepad
- Wait for the window "Untitled – Notepad"
- Send our lines of text
- Press ALT-FA
- Wait for the window "Save As"
- Send our filename
- Press Enter

So let's turn this into a Macro Scheduler script. Read through the following sections in turn.

Run Notepad 44
Wait for the window "Untitled – Notepad" 44
Send our lines of text 44
Press ALT-FA 46
Wait for the window "Save As" 46
Send our filename 46
The Script So Far 48
Printing the Document 49
Closing Notepad 50

© 2015 MJT Net Ltd

3.6.1 Run Notepad

There are a number of ways to run Notepad. I expect most of us will use our mouse to click on the Start button, and find the shortcut in our Programs list. All this does is execute a shortcut, which in turn runs Notepad (more technically, it runs the Notepad executable – notepad.exe). Since the shortcut could appear anywhere and no doubt appears in different places for different people depending on their preferences, and since all it does is run Notepad, we should bypass the shortcut and run Notepad directly, in the same way that we would if using the Run command in Windows Start menu.

Furthermore, as the Notepad program lives in the System folder, Windows knows where to find it so we don't even have to specify the path to it.

So in Macro Scheduler we just have to write the following line:

Run>Notepad.exe

In many other automation scenarios, however, this won't be the case, and sometimes it is necessary to switch to an application's directory before running the application. In which case we would do something like this:

Change Directory>c:\windows\system32
Run>c:\windows\system32\notepad.exe

This is totally unnecessary for Notepad, but there's no harm demonstrating how you might have to execute your own applications.

3.6.2 Wait for the window "Untitled – Notepad"

Macro Scheduler gives us two very useful commands: WaitWindowOpen and WaitWindowClosed.

As their names suggest these wait for the specified window to be fully open, or closed respectively before script execution continues. The argument given to these commands can be the exact window title, or a portion of the window title followed by an asterisk. The latter tells Macro Scheduler to search all windows until it finds a window whose title *contains* the specified text regardless of case and stop at the first one it finds.

In our case we know the title is always "Untitled - Notepad" after running Notepad, so we can write the next line of our script:

WaitWindowOpen>Untitled - Notepad

3.6.3 Send our lines of text

For this we use the SendText command, commonly abbreviated to just Send:

Send>some text

To send more than one line we would manually type the text and press enter to move on to the next line, so this would look like this:

Send>This is the first line of my notepad file
Press Enter

© 2015 MJT Net Ltd

Send>This is line two
Press Enter
Send>This is the third and final line.

But let's be a bit cleverer. Let's create three lines of text and assign them to variables using the Let command:

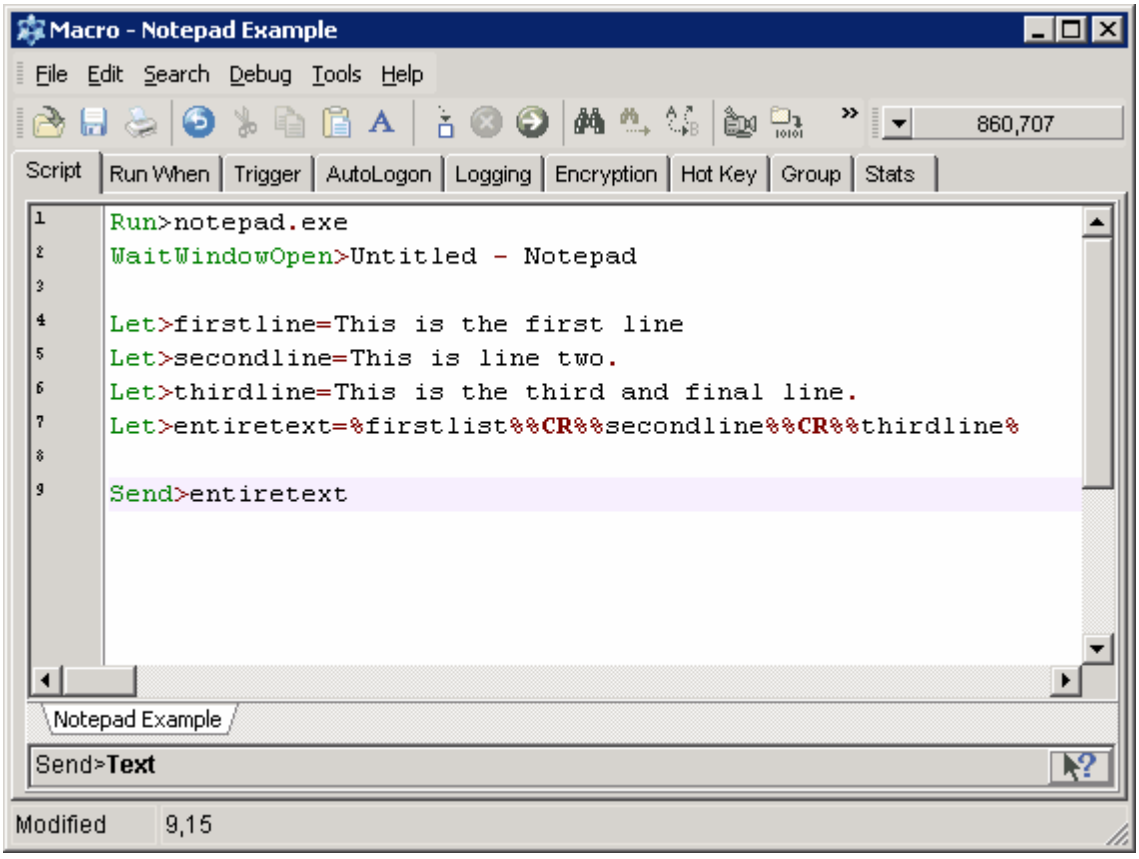
Let>firstline=This is the first line of my notepad file.
Let>secondline=This is line two.
Let>thirdline=This is the third and final line.

Now let's join them together with a carriage return character to ensure the line breaks. CR is a Macro Scheduler system variable for a carriage return:

Let>entiretext=%firstline%%CR%%secondline%%CR%%thirdline%
Send>entiretext

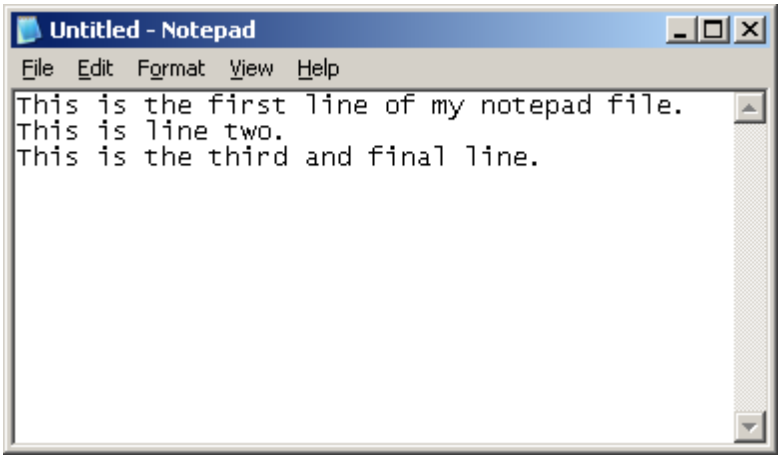
The % symbols tell Macro Scheduler that the value within the % symbols is a variable, so it creates a new variable called 'entiretext' made up of the variables firstline, CR, secondline, CR and thirdline all joined together.

Let's test the script so far. Close any open Notepad windows and run the script:



Hopefully you ended up with a new Notepad window appearing on your desktop looking like this:

© 2015 MJT Net Ltd


Untitled - Notepad
File Edit Format View Help
This is the first line of my notepad file.
This is line two.
This is the third and final line.

3.6.4 Press ALT-FA

When we press the ALT key we will hold it down, send the other keys with it and then release the ALT key again. So we actually do this:

Press ALT
Send the F and A characters
Release ALT

In Macro Scheduler this is almost exactly as written above:

Press ALT
Send>fa
Release ALT

I prefer to use lower case characters when I am issuing keystrokes. It sometimes avoids issues where an upper case character can be interpreted as having the Shift key down at the same time – think about it ...

Remember the Release ALT. With the ALT, SHIFT and CTRL keys we should always remember to release them after pressing them. They are used in conjunction with other keys. If we forget to Release the ALT key, subsequent key sends will act as if the ALT key was also pressed because it has yet to be released.

3.6.5 Wait for the window "Save As"

We've dealt with this before, so without further ado:

WaitWindowOpen>Save As

3.6.6 Send our filename

When the Save As dialog appears, the filename box has the keyboard focus, so all we need to do is send our filename, and it will be filled in the right place. If we needed to move keyboard focus a good tip is to use Press Tab to move from object to object.

What we should do is set a variable at the top of the script with our filename, so that we can modify that easily:

© 2015 MJT Net Ltd

Let>filename=c:\my documents\sample.txt

Assuming we have done this we could just now send the filename and press enter to save it as follows:

Send>filename
Press Enter

However, what if this file already exists? We would get an error and we would have to script the error window and make changes as necessary.

It would be easier to avoid this and check the existence of the file first. Macro Scheduler has a command called IfFileExists. But what do we do if it does exist? We could just delete it? This is fine in this instance because I know that this file is just an example and I'm never going to confuse it with a valid alternative file. But if we wanted to play safe we could give it a new filename. We could use the date and time, and/or a random number on the filename and keep performing the existence check before saving the file.

To keep things simple at this stage we'll just delete the file if it already exists. IfFileExists is given the filename and proceeds onto the next line if it exists. If it doesn't exist it jumps to the associated Endif line. So we should do this:

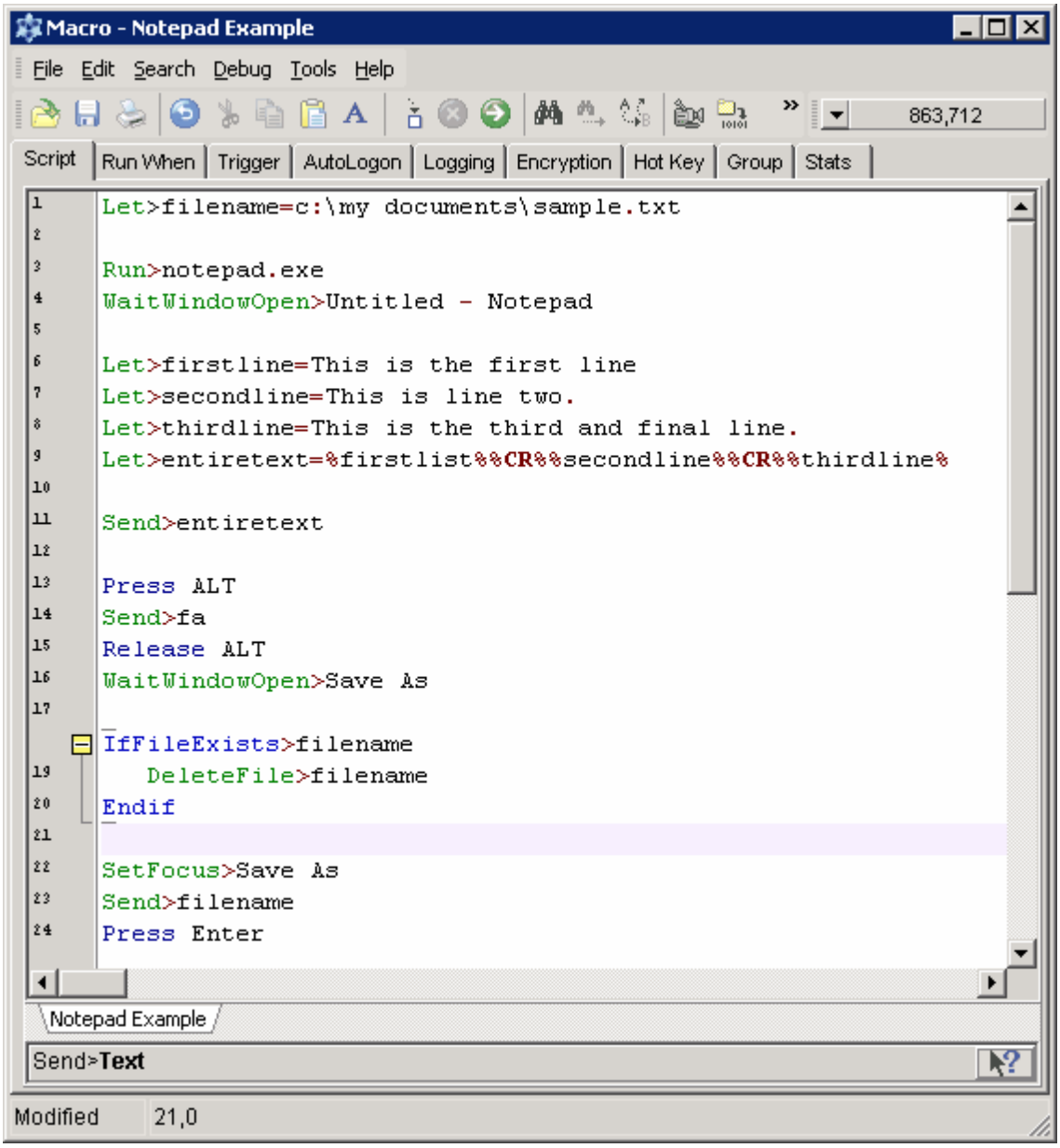
IfFileExists>filename
* DeleteFile>filename*
Endif

SetFocus>Save As
Send>filename
Press Enter

The last two lines here are the two that send the filename and press enter to save it. We've added a SetFocus line just for good measure. It shouldn't be needed as the Save As dialog should already be active, but for the sake of example it has been added. There is never any harm in ensuring the correct window has the focus before we send any keystrokes.

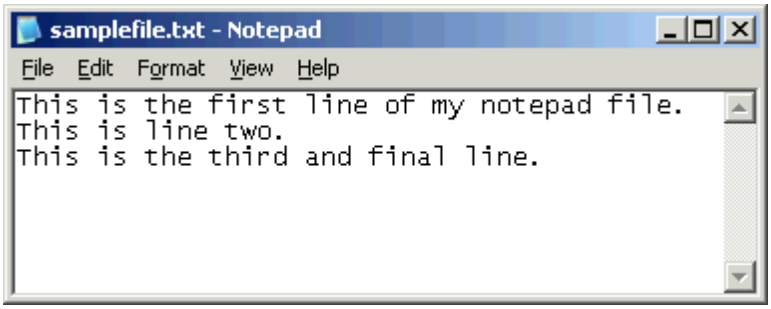
© 2015 MJT Net Ltd

3.6.7 The Script So Far



Note the variable 'filename' declared at the top.

After running the script you should see a few flashes as the windows and dialogs appear and disappear faster than you can type and the end result should look something like:


samplefile.txt - Notepad
File Edit Format View Help
This is the first line of my notepad file.
This is line two.
This is the third and final line.

Note the new window title – "samplefile.txt – Notepad" – proof that the file has been saved.

© 2015 MJT Net Ltd

If you close Notepad and run the script again the same thing will happen. No errors, because the filename has been deleted and there was no clash.

3.6.8 Printing the Document

Having worked through sending keystrokes to initiate the Save As dialog, printing shouldn't be very much different. If we look at the File menu we can see that 'P' does the print. In later versions of Windows Notepad CTRL-P is also a shortcut for printing, so we can use either ALT-FP or CTRL-P.

Remember it is always sensible to focus the app we want to send keystrokes to. The Notepad window title has changed – it is no longer 'Untitled – Notepad' as we have saved the document.

We could give the new window title, but what if we can't predict easily what the filename is going to be? Assuming no other Notepad windows are open we can use the wildcard as explained previously when discussing WaitWindowOpen. So we can do this:

*SetFocus>notepad**
Press ALT
Send>fp
Release ALT

What happens when we send ALT-FP to Notepad depends on which version of Notepad you have. In earlier versions Notepad just went straight ahead and printed the document. But in later versions it displayed the Print dialog. So please try it manually first to decide which version you have. This reiterates the importance of working through the process manually to understand what needs to be automated.

If the Print dialog does not appear and Notepad just starts printing the document then skip directly to Step 2.

Step 1

If the Print dialog appears, we should wait for it to appear – it's window title is 'Print' so we would do this:

WaitWindowOpen>Print

Now, we just want to print to the default printer. As it is the default it is already selected and we just have to press enter to activate the default 'Print' button:

Press Enter

And the Print dialog now disappears.

Step 2

What happens next? Well, if you look closely you will see that a new, smaller dialog, entitled 'Notepad' appears indicating that it is printing. With the small document we are dealing with here it will flash up rather quickly, but we need to be aware of it. Our macro needs to wait for this to disappear before we can finally close Notepad.

So after pressing enter on the print dialog we should wait for this little status window to appear, and then disappear again before we know Notepad is ready again for input. We can do this with a WaitWindowOpen followed by a WaitWindowClosed:

© 2015 MJT Net Ltd

WaitWindowOpen>Notepad
WaitWindowClosed>Notepad

Don't be tempted to just use a WaitWindowClosed command on its own, as this pauses until the specified window is not present, and immediately after pressing Enter, the window may not yet be present – your script's so fast that the little dialog has not had a chance to appear – so the script would continue. We should wait for it to open and then wait for it to disappear. This is the only sure-fire method.

3.6.9 Closing Notepad

So we've printed the document. Now all we need to do is close Notepad. How do we close Notepad with keystrokes? There are a number of ways. Notice that under the File menu is the usual Exit option. If we use ALT-F to open the file menu the Exit option has 'x' underlined, so ALT-FX will close Notepad.

In Windows ALT-F4 will also close a window. Macro Scheduler also offers a CloseWindow command, which will probably do the job (though this doesn't necessarily close all applications as not all programs process "close window" messages in the same way).

Let's use ALT-FX to simulate true user input:

*SetFocus>Notepad**
Press ALT
Send>fx
Release ALT

And Notepad should disappear.

3.7 The Complete Script

So let's review the script. I have added comments to document the script – to make it more readable and easy to follow. You can use almost anything that isn't a recognised script command for comments. There is a reserved word called Remark which can be used:

Remark>This is a comment

But I've used // to indicate my comments. Some people like to use **. Use what you feel comfortable with – as long as it isn't a recognised script command – or it will try to execute your comments. Which could be interesting!

Remember to modify the section of the script that does the printing depending on whether or not your Notepad displays the Print dialog. There are clever ways that we could make the script generic to cope with both cases, but that's for a later discussion. But if you're interested look at the WW_TIMEOUT variable that you can set for WaitWindowOpen, and the IfWindowOpen command.

If you have any problems running this script you may want to slow it down a bit. We have made it run as fast as it possibly can, but some systems react too slowly for windows to be ready in time for the script. Consider adding Wait statements at key places, such as before a SetFocus, or before sending keystrokes:

//This will wait 2 seconds.
Wait>2

© 2015 MJT Net Ltd

```
//Set the name of our file
Let>filename=c:\samplefile.txt

//Start Notepad
Run>notepad.exe
WaitWindowOpen>Untitled - Notepad

//Construct the text
Let>firstline=This is the first line of my notepad file.
Let>secondline=This is line two.
Let>thirdline=This is the third and final line.
Let>entiretext=%firstline%%CR%%secondline%%CR%%thirdline%

//Send the text to Notepad
Send>entiretext

//Save As
Press ALT
Send>fa
Release ALT
WaitWindowOpen>Save As

//Delete the file if it already exists
IfFileExists>filename
   DeleteFile>filename
Endif

//Send the filename to the Save As dialog
SetFocus>Save As
Send>filename
Press Enter

//Print the file
SetFocus>notepad*
Press CTRL
Send>p
Release CTRL

//Remove the next two lines if your version of
//Notepad does not display the Print dialog
WaitWindowOpen>Print
Press Enter

//Wait for the print status dialog to finish
WaitWindowOpen>Notepad
WaitWindowClosed>Notepad

//Now close Notepad
SetFocus>Notepad*
Press ALT
Send>fx
Release ALT
```

© 2015 MJT Net Ltd

3.8 Where To Go From Here

In this guide we have met the following common commands:

Run Program
WaitWindowOpen
WaitWindowClosed
SetFocus
Send Character/Text (Send)
Press ...
Label and Goto
Wait

These commands will always be found in Windows automation scripts. They are essential for simulating user input, and we have seen how just these commands can be used to automate a Windows application. Other useful commands for windows manipulation include the mouse event functions:

MouseMove
MouseMoveRel
LClick
LDblClick
RClick
RDblClick

For further detail on these commands please refer to the Command Reference.

The Command Reference also gives examples for each command.

Further scripting articles and tutorials can be found at http://www.mjtnet.com/blog and there are tips, examples and lots of helpful people in the forums at http://www.mjtnet.com/usergroup/

You'll also find video tutorials at: http://www.mjtnet.com/video_tutorials.htm

Happy Scripting!

© 2015 MJT Net Ltd

Command Reference

Part IV

4 Command Reference

4.1 Complex Expressions

Syntax

In Macro Scheduler all complex functions should be contained within curly braces ("{" and "}"). Variables should be enclosed within % symbols and literal strings should be enclosed within double quotes ("). Parameters are separated by commas (,).

Complex expressions are supported in Macro Scheduler's IF 146 statements and LET 203 statement, and since version 9.0 can now also be included within all other commands and function calls in the place of regular variables.

Complex expressions can be disabled at any point by setting the DISABLE_COMPLEX_EXPRESSIONS variable to 1. Switch back on by resetting to 0.

Operators

Arithmetic Operators 54
Logical Operators 55
String Operators 55
Relational Operators 55

Functions

Arithmetic Functions 55
String Functions 56
Conditional Expressions 57

Examples

Complex Expression Examples 57

4.1.1 Arithmetic Operators

Binary arithmetic operators:

^	exponent
+	addition
-	subtraction
*	multiplication
/	division
div	integer division
mod	remainder

Unary arithmetic operators:

+	sign positive identity
-	sign negation

© 2015 MJT Net Ltd

4.1.2 Logical Operators

Bitwise logical operators

not	bitwise negation
and	bitwise and
or	bitwise or
xor	bitwise xor
shl	shift left
shr	shift right

Boolean logical operators

not	negation
and	logical and
or	logicalor
xor	logical xor

4.1.3 String Operators

String operators

+	concatenation

4.1.4 Relational Operators

=	equal
<>	not equal
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

4.1.5 Arithmetic Functions

Examples 57

function Trunc(X: Extended):Integer
The Trunc function truncates a float-type value to an integer-type value. X is a float-type expression. Trunc returns an Integer value that is the value of X rounded toward zero.

function Round(X: Extended): Integer
The Round function rounds a float-type value to an integer-type value. X is a float-type expression. Round returns a Longint value that is the value of X rounded to the nearest whole number using "bankers rules" where exact halfs are rounded to the nearest even number.

function Abs(X): Float
The Abs function returns the absolute value of the argument. X is an integer-type or float-type expression.

function ArcTan(X: Float): Float
ArcTan calculates the arctangent of the given number. Calculate other trigonometric functions using Sin, Cos, and ArcTan in the following expressions
Tan(x) = Sin(x) / Cos(x)
ArcSin(x) = ArcTan (x/sqrt (1-sqr (x)))
ArcCos(x) = ArcTan (sqrt (1-sqr (x)) /x)

© 2015 MJT Net Ltd

function Cos(X: Float): Float
The Cos function returns the cosine of the angle X, in radians.

function Exp(X: Float): Float
Exp returns the value of e raised to the power of X, where e is the base of the natural logarithms.

function Frac(X: Float): Float
The Frac function returns the fractional part of the argument X. X is a float-type expression. The result is the fractional part of X; that is: Frac(X) = X - Int(X).

function Int(X: Float): Float
X is a float-type expression. The result is the integer part of X; that is, X rounded toward zero.

function Ln(X: Float): Float
The Ln function returns the natural logarithm (Ln(e) = 1) of the float-type expression X.

function Pi: Float
Use Pi in mathematical calculations that require pi, the ratio of a circle's circumference to its diameter. Pi is approximated as 3.1415926535897932385.

function Sin(X: Float): Float
The Sin function returns the sine of the argument. X is a float-type expression. Sin returns the sine of the angle X in radians.

function Sqr(X: Float): Float
The Sqr function returns the square of the argument. X is a floating-point expression. The result, of the same type as X, is the square of X, or X*X.

function Sqrt(X: Float): Float
X is a floating-point expression. The result is the square root of X.

function Power(Base, Exponent: Float): Float
The Power function raises Base to any power. For fractional exponents or exponents greater than MaxInt, Base must be greater than 0.

4.1.6 String Functions

Examples 57

function Upper(S: string): string
The Upper function returns a string containing the same text as S, but with all 7-bit ASCII characters between 'a' and 'z' converted to uppercase.

function Lower(S: string): string
Lower returns a string with the same text as the string passed in S, but with all letters converted to lowercase. The conversion affects only 7-bit ASCII characters between 'A' and 'Z'.

function Copy(S: string; Index, Count: Integer): string
The Copy function returns a substring of a string. S is a string-type expression. Index and Count are integer-type expressions. Copy returns a string containing Count characters starting at S[Index]. If Index is larger than the length of S, Copy returns an empty string. If Count specifies more characters than are available, the only the characters from S[Index] to the end of S are returned.

function Pos(Substr: string; S: string): Integer
Pos searches for a substring, Substr, in a string, S. Substr and S are string-type expressions. Pos

© 2015 MJT Net Ltd

searches for Substr within S and returns an integer value that is the index of the first character of Substr within S. Pos ignores case-insensitive matches. If Substr is not found, Pos returns zero.

function Length(S: string): Integer
The Length function returns the number of characters actually used in the string S.

function Trim(S: string): Integer
The Trim function returns the string with any leading and trailing spaces and control characters removed.

function LTrim(S: string): Integer
The LTrim function returns the string with any leading spaces and control characters removed.

function RTrim(S: string): Integer
The RTrim function returns the string with any trailing spaces and control characters removed.

4.1.7 Conditional Functions

Examples 57

function If(Condition: Boolean, TrueResult, FalseResult): ResultType
Condition is a Boolean expression. When the function is evaluated, it returns TrueResult if Condition else FalseResult. TrueResult and FalseResult need not be of the same type and result type of the IF expression may change depending on Condition.

4.1.8 Examples

Numeric Functions:

```
Let>rounded={round(34.6)}
Let>ans={cos(340)}
```

Using numeric variables in complex expressions we enclose variables in % symbols:

```
Let>decimal=34.6
Let>rounded={round(%decimal%)}
```

String Functions:

In Complex expressions string literals must be enclosed in double quotes ("). Variables are just enclosed with % symbols:

```
Let>email=fred@someserver.com
Let>at={Pos("@",%email%)}
```

Here at equals 5 as it is the 5th character in fred@someserver.com

Let's now use the copy function to extract everything before the @ sign:

```
Let>namepart={copy(%email%,1,%at%-1)}
```

Conditional Functions:

```
Let>email=fred@someserver.com
Let>IsItAnEmail={if(pos("@",%email%)>0,"yes","no")}
```

© 2015 MJT Net Ltd

Of course we could do:

```
Let>email=fred@someserver.com
Let>at={Pos("@",%email%)}
Let>IsItAnEmail={if(%at%>0,"yes","no")}
```

Here IsItAnEmail would be set to "yes".

4.2 Clipboard Commands

4.2.1 GetClipBoard

GetClipBoard>result_variable[,format]

Retrieves the contents of the clipboard as text and places it in the specified variable.

The optional format parameter can be used to retrieve the text in a different format as follows:

0: CF_TEXT (Default)
1: CF_HTML - if the clipboard contains HTML the text raw HTML will be returned.

Abbreviation : GCB
See also: PutClipBoard 58, WaitClipBoard 58

Example

```
GetClipBoard>WhatsInTheClipBoard
```

4.2.2 PutClipBoard

PutClipBoard>SomeValue

Places the specified text onto the clipboard. A variable may be used, or a literal value.

Abbreviation : Put
See also: GetClipBoard 58, WaitClipBoard 58

Example

```
PutClipBoard>Hello World !!
```

4.2.3 WaitClipBoard

WaitClipBoard

Waits for the clipboard to be ready to access (not open by another application).

Abbreviation: WCB
See also: GetClipBoard 58, PutClipBoard 58

© 2015 MJT Net Ltd

4.3 Console App Funtions

4.3.1 SOWrite

SOWrite>text

Writes text to STDOUT. For compiled console EXEs only. This function does nothing in non-compiled scripts or compiled scripts that have not been compiled as console applications.

Abbreviation: SOW
See also: SOWriteLn 59

4.3.2 SOWriteLn

SOWriteLn>text

Writes line of text to STDOUT. For compiled console EXEs only. This function does nothing in non-compiled scripts or compiled scripts that have not been compiled as console applications.

Abbreviation: SOL
See also: SOWrite 59

4.4 Database Commands

4.4.1 DBClose

Not supported in Macro Scheduler Lite.

DBClose>database_reference

Closes a database connection established by the DBConnect command.

Abbreviation: DBX
See also: DBConnect 60, DBExec 60, DBQuery 61

For more advanced database manipulation use VBScript. See:
http://www.mjtnet.com/blog/2006/02/20/accessing-databases/

Example

```
Let>str=Driver={MySQL ODBC 3.51
Driver};Server=someserver.com;Port=3306;Database=example;User=admin;Password=xxxx;O
ption=3;
DBConnect>str,dbH

Let>SQL=delete from mytable where ID=1234
DBExec>dbH,SQL,result

DBClose>dbH
```

© 2015 MJT Net Ltd

4.4.2 DBConnect

DBConnect>connection_string,database_reference

Not supported in Macro Scheduler Lite.

Connects to a database via a system data source or ADO/ODBC connection string.

Accepts an ADO/ODBC connection string or system data source name.

Returns a reference to the database, to be used in DBClose, DBExec and DBQuery commands.

If an error occurs database_reference will be set to "ERR" followed by any error message returned by ADO or the database driver.

Use DBClose to close the database.

To set a timeout for the connection set the DB_CONNECTTIMEOUT variable to a number of seconds.

For more advanced database manipulation use VBScript. See:
http://www.mjtnet.com/blog/2006/02/20/accessing-databases/

Abbreviation: DBC
See also: DBClose 59, DBExec 60, DBQuery 61

Example

```
Let>str=Driver={MySQL ODBC 3.51
Driver};Server=someserver.com;Port=3306;Database=example;User=admin;Password=xxxx;O
ption=3;
DBConnect>str,dbH

Let>SQL=delete from mytable where ID=1234
DBExec>dbH,SQL,result

DBClose>dbH
```

4.4.3 DBExec

Not supported in Macro Scheduler Lite.

DBExec>database_reference,SQL_Statement,result

Executes an SQL statement. Use DBExec to execute SQL statements that do not return a recordset. E.g. DELETE, INSERT and UPDATE queries.

For SELECT queries use DBQuery to return a recordset array.

Accepts a database reference created by a call to DBConnect, and an SQL_Statement. Specify a result variable to return the number of rows affected by the SQL statement.

To set a timeout for the command set the DB_COMMANDTIMEOUT variable to a number of seconds.

© 2015 MJT Net Ltd

For more advanced database manipulation use VBScript. See:
http://www.mjtnet.com/blog/2006/02/20/accessing-databases/

Abbreviation: DBE
See also: DBConnect 60, DBClose 59, DBQuery 61

Example

```
Let>str=Driver={MySQL ODBC 3.51
Driver};Server=someserver.com;Port=3306;Database=example;User=admin;Password=xxxx;O
ption=3;
DBConnect>str,dbH

Let>SQL=delete from mytable where ID=1234
DBExec>dbH,SQL,result

DBClose>dbH
```

4.4.4 DBQuery

Not supported in Macro Scheduler Lite.

DBQuery>database_reference,SQL_Statement,recordset_array,num_recs,num_fields[,fieldnames]

Use DBQuery to perform a SQL query which returns a recordset, such as a SELECT statement.

database_reference is a reference variable returned by a previous call to DBConnect. Specify the SQL statement in SQL_Statement.

Specify a variable name in recordset_array in which to return the recordset. The array is constructed in the format recordset_rownum_fieldnum. So a recordset with 2 rows and 3 fields per row would create an array as follows:

variablename_1_1
variablename_1_2
variablename_1_3
variablename_2_1
variablename_2_2
variablename_2_3

Specify variables for num_recs and num_fields to store the number of records and number of fields returned in the recordset.

Optionally set fieldnames to 1 to return the names of the fields rather than the index. This would return an array like:

variablename_1_FIELD1
variablename_1_FIELD2
variablename_1_FIELD3
variablename_2_FIELD1
variablename_3_FIELD2
variablename_3_FIELD3

Where FIELD1, FIELD2, FIELD3 were the actual field names of the fields returned.

© 2015 MJT Net Ltd

To set a timeout for the command set the DB_COMMANDTIMEOUT variable to a number of seconds.

For more advanced database manipulation use VBScript. See:
http://www.mjtnet.com/blog/2006/02/20/accessing-databases

Abbreviation: DBQ
See also: DBConnect 60, DBClose 59, DBExec 60

Example

```
//Connect to Datasource
Let>str=Driver={MySQL ODBC 3.51 Driver};Server=someserver.com;Port=3306;Database=example;User=a
DBConnect>str,dbH

//Perform SELECT query
Let>SQL=select * from customers where custID='abc123'
DBQuery>dbh,SQL,CUSTOMERS,numrecs,numfields

//loop through returned recordset
Let>r=0
Repeat>r
  Let>r=r+1
  Let>f=0
  Repeat>f
    Let>f=f+1
    Let>this_field=CUSTOMERS_%r%_%f%
    Message>this_field
    Wait>0.5
  Until>f=numfields
Until>r=numrecs

//Close database connection
DBClose>dbH
```

4.5 Date/Time Commands

4.5.1 DateAdd

DateAdd>date,interval,amount,result

Adds (or subtracts) a number of days, weeks, months or years to (or from) a date.
Interval can take one of the following values:

D: Days
W: Weeks
M: Months
Y: Years

Amount can be a positive or negative amount to apply to the date.

The date must be in local system format. Use DateLocal 63 to generate a date in local system

© 2015 MJT Net Ltd

format.

See also: DateDiff 63, DateLocal 63, DatePart 63

Examples

```
GetDate>today
DateAdd>today,D,-1,yesterday
DateAdd>today,M,1,NextMonth
DateAdd>today,W,-1,LastWeek
DateAdd>today,Y,-2,TwoYearsAgo
```

4.5.2 DateDiff

DateDiff>date1,date2,interval,result

Returns the difference between two dates. Interval can be one of:

D: Days
W: Weeks
M: Months
Y: Years

The dates must be in local system format. Use DateLocal 63 to generate a date in local system format.

See also: DateAdd 62, DateLocal 63, DatePart 63

Example:

```
GetDate>today
DateAdd>today,M,1,NextMonth

DateDiff>today,NextMonth,D,numberDays
```

4.5.3 DateLocal

DateLocal>Year,Month,Day,result

Given year, month and day portions DateLocal returns a date in local system format.

See also: DateAdd 62, DateDiff 63, DatePart 63

Example:

```
DateLocal>2014,03,21,theDate
```

4.5.4 DatePart

DatePart>date,interval,result

Returns part of a date specified by interval. Interval can be one of:

© 2015 MJT Net Ltd

D: Day of month
M: Month
Y: Year

The date should be in a format recognised by the system. Use DateLocal 63 to generate a date in local system format.

See also: DateAdd 62, DateDiff 63, DateLocal 63

Example

```
GetDate>today
DatePart>today,M,thisMonth
```

4.5.5 DateStamp

DateStamp>filename,comment

Outputs the date and time in milliseconds and the given comment to the given text file.

Abbreviation : DAT
See also: WriteLn 128, TimeStamp 69

Example

```
DateStamp>c:\temp\mylogfile.txt,Macro Started
```

The file entry would appear :

1999-08-06:13:38:10:352 - Macro Started

4.5.6 Day

Day>result

Returns the current day number of the month in the specified variable.

See also: Month 67, Year 69, GetDate 65, GetTime 66

Example

```
Day>the_day
Month>the_month
Year>the_year
Message>The date is : %the_day% - %the_month% - %the_year%
```

4.5.7 DayOfWeek

DayOfWeek>result

Returns the current week day number, starting with Sunday as day 1, and ending on Saturday with

© 2015 MJT Net Ltd

day 7.

Abbreviation : DOW
See also: Day 64, Month 67, Year 69, GetDate 65, GetTime 66

Example

This example displays the current day as a proper day name. It also shows how to use variables in a Goto command.

```
DayOfWeek>result
Goto>Day%result%

Label>Day1
Let>DayString=Sunday
Goto>Continue

Label>Day2
Let>DayString=Monday
Goto>Continue

Label>Day3
Let>DayString=Tuesday
Goto>Continue

Label>Day4
Let>DayString=Wednesday
Goto>Continue

Label>Day5
Let>DayString=Thursday
Goto>Continue

Label>Day6
Let>DayString=Friday
Goto>Continue

Label>Day7
Let>DayString=Saturday
Goto>Continue

Label>Continue
MessageModal>DayString
```

4.5.8 GetDate

GetDate>result

Returns the current date in the specified variable. The format of the date depends on the regional settings of the system.

Abbreviation : GDT
See also: GetTime 66, Year 69, Month 67, Day 64, FileDate 121

Example

```
GetDate>date
Let>msg=The Date Is :
```

© 2015 MJT Net Ltd

```
ConCat>msg,date
Message>msg
```

4.5.9 GetTime

GetTime>result

Returns the current time in the specified variable. The format of the time depends on the regional settings of the system.

Abbreviation : GTM
See also: GetDate 65, Year 69, Month 67, Day 64, FileDate 121

Example

```
GetTime>time
Let>msg=The Time Is :
ConCat>msg,time
Message>msg
```

4.5.10 Hour

Hour>result

Returns the hour portion of the current time in the specified variable.

See also: Min 66, Sec 67, GetDate 65, GetTime 66

Example

```
Sec>Seconds
Min>Minutes
Hour>Hour
Message>The time is : %Hour% : %Minutes% : %Seconds%
```

4.5.11 Min

Min>result

Returns the minutes portion of the current time in the specified variable.

See also: Sec 67, Hour 66, GetDate 65, GetTime 66

Example

```
Sec>Seconds
Min>Minutes
Hour>Hour
Message>The time is : %Hour% : %Minutes% : %Seconds%
```

© 2015 MJT Net Ltd

4.5.12 Month

Month>result

Returns the current month number in the specified variable.

Abbreviation : Mon
See also: Day 64, Year 69, GetDate 65, GetTime 66

Example

```
Day>the_day
Month>the_month
Year>the_year
Message>The date is : %the_day% - %the_month% - %the_year%
```

4.5.13 Sec

Sec>result

Returns the seconds portion of the current time in the specified variable.

See also: Min 66, Hour 66, GetDate 65, GetTime 66

Example

```
Sec>Seconds
Min>Minutes
Hour>Hour
Message>The time is : %Hour% : %Minutes% : %Seconds%
```

4.5.14 TimeAdd

TimeAdd>time,interval,amount,result

Adds (or subtracts) a number of hours, minutes or seconds to (or from) a time.

Interval can take one of the following values:

H: Hours
M: Minutes
S: Seconds

Amount can be a positive or negative amount to apply to the time.

The time must be in local system format. Use TimeLocal 68 to generate a time in local system format.

See also: TimeDiff 68, TimeLocal 68, TimePart 68

Examples

```
GetTime>now
TimeAdd>now,H,-1,OneHourEarlier
```

© 2015 MJT Net Ltd

```
TimeAdd>now,M,30,HalfHourLater
```

4.5.15 TimeDiff

TimeDiff>time1,time2,interval,result

Returns the difference between two times. Interval can be one of:

H: Hours
M: Minutes
S: Seconds

The times must be in local system format. Use TimeLocal 68 to generate a time in local system format.

See also: TimeAdd 67, TimeLocal 68, TimePart 68

Example:

```
GetTime>now
TimeAdd>now,H,12,LaterTime

TimeDiff>now,LaterTime,M,numberMins
```

4.5.16 TimePart

TimePart>time,interval,result

Returns part of a time specified by interval. Interval can be one of:

H: Hour
M: Minute
S: Second

The time should be in a format recognised by the system. Use TimeLocal 68 to generate a time in local system format.

See also: TimeAdd 67, TimeDiff 68, TimeLocal 68

Example

```
GetTime>now
TimePart>now,H,CurrentHour
```

4.5.17 TimeLocal

TimeLocal>Hour,Minute,Second,Result

Given hour (24 hour notation), minute and second portions TimeLocal returns a time in local system format.

© 2015 MJT Net Ltd

See also: TimeAdd 67, TimeDiff 68, TimePart 68

Example:

```
TimeLocal>13,15,00,localTime
```

4.5.18 Timer

Timer>result

Returns the number of milliseconds that have elapsed since the script was started.

Example:

```
Timer>startTime
..
.. do something here
..
Timer>endTime
Let>elapsed_seconds={(%endTime%-%startTime%)/1000}
MessageModal>Seconds elapsed: %elapsed_seconds%
```

4.5.19 TimeStamp

TimeStamp>filename,comment

Outputs the time in milliseconds and the given comment to the given text file.

Abbreviation : TIM
See also: WriteLn 128, DateStamp 64

Example

```
TimeStamp>c:\temp\mylogfile.txt,Macro Finished
```

The file entry would appear :

15:43:03:066 - Macro Finished

4.5.20 Year

Year>result

Returns the current year in the specified variable.

See also: Month 67, Day 64, GetDate 65, GetTime 66

Example

```
Day>the_day
Month>the_month
Year>the_year
Message>The date is : %the_day% - %the_month% - %the_year%
```

© 2015 MJT Net Ltd

4.6 DDE Commands

4.6.1 DDEPoke

DDEPoke>Server,Topic,Item,Data

Pokes data to the given DDE server. The server's DDE topic and item must be specified.

Abbreviation : DPK
See also: DDERequest 70

Example

```
DDEPoke>MyServer,System,Item1,Testing 123
```

4.6.2 DDERequest

DDERequest>Server,Topic,Item,Result,Timeout

Requests data from a DDE server. The data returned is stored in the Result variable. If the conversation does not complete within the Timeout value specified, Result will contain 'DDE_TIMEDOUT'. The Timeout value is in seconds.

Abbreviation : DRQ
See also: DDEPoke 70

Example

The following example gets the URL and window title from Netscape.

```
DDERequest>Netscape,WWW_GetWindowInfo,0xFFFFFFFF,ret,10
Message>ret
```

The DDERequest command can also be used to open a web page in Netscape :

```
DDERequest>Netscape,WWW_OpenUrl,www.mjtnet.com,ret,0
```

4.7 Dialogs

Custom dialogs can be used to create interactive macros which request information from or present information to the user. Dialogs can display all kinds of objects and their appearance can be fully customised. Event handlers can be created to respond to user input.

Macro Scheduler's custom dialogs are designed for basic user interaction. The aim of custom dialogs is to give the programmer the ability to create macros which request information from the user. They are NOT designed for creating full featured user interfaces and do not offer the same amount of interface objects and flexibility provided by, for example, Visual Basic.

Dialog Functions 101
Dialog Objects 71
Modal vs Non-Modal Dialogs 107

© 2015 MJT Net Ltd

Examples 107

4.7.1 Dialog Objects

The following types of object are currently supported:

Form (The Dialog Itself)
Label 76
Edit 78
Memo 79
Button 72
CheckBox 81
ListBox 83
ComboBox 85
Image 87
RadioGroup 88
ProgressBar 90
PageControl 91
TabSheet 93
Panel 94
Splitter 96
StatusBar 97
StringGrid 98
MenuItem 100
HTMLViewer 101

4.7.1.1 File Browse Buttons

Button objects can invoke standard file Open and Save dialogs as well as the Folder Browse dialog. The following properties exist:

BrowseStyle
Filename
InitialDir
Filter
DoBrowse

BrowseStyle can take one of the following values to make the button initiate one of the following dialog types:

fbOpen: File Open Dialog
fbSave: File Save Dialog:
fbFolder: File Browse Dialog

InitialDir should be used to set the initial folder for the dialog, and Filter the file spec filter (e.g. *.txt). Filename can both be set with an initial filename and also is used to get the filename chosen by the user.

Setting DoBrowse to True invokes the browse operation.

E.g. The following code causes the DoBrowse subroutine to be executed when the user clicks a button. Inside DoBrowse we then invoke the button's file browse dialog.

//Set an Event Handler for the button's OnClick event

© 2015 MJT Net Ltd

```
AddDialogHandler>Dialog1,btnBrowse,OnClick,DoBrowse


//Here we perform the file browse, then get the filename and output it to an edit box
SRT>DoBrowse
  SetDialogProperty>Dialog1,btnBrowse,DoBrowse,True
  GetDialogProperty>Dialog1,btnBrowse,Filename,str_Filename
  SetDialogProperty>Dialog1,txtFilename,Text,str_Filename
END>DoBrowse
```

4.7.1.2 Button Object Properties

Name	Description
Action	Designates the action associated with the control.
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BiDiMode	Specifies the bi-directional mode for the control.
BitmapData	Bitmap data for data imported into macro. Use LabelToVariable to get a variable to set this property to.
BrowseStyle	Determine whether to use Open File/Save As or Folder Browse dialog for DoBrowse option.
Cancel	Determines whether the button's OnClick event handler executes when the Escape key is pressed.
Caption	Specifies a text string that identifies the control to the user.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
Default	Determines whether the button's OnClick event handler executes when the Enter key is pressed.
DoBrowse	Set to True to initiate a file open/save as/folder browse dialog. Set BrowseStyle for the type of dialog to use.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Filename	Returns the filename located via the DoBrowse option.
Filter	Directory filter for DoBrowse dialog.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
InitialDir	Initial directory for DoBrowse operation.
Kind	Specifies the kind of bitmap button.
Layout	Specifies where the image appears on the bitmap button.

© 2015 MJT Net Ltd

Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
LoadBitmap	Specify a filename of an image to load onto button. Right click in properties grid to select "File Browse" option to locate a bitmap file. Note that the image data is loaded statically into and stored in the dialog. To link dynamically assign a filename to LoadBitmap at runtime.
Margin	Specifies the number of pixels between the edge of the image (specified in the Glyph property) and the edge of the button.
Margins	Specifies the margins for the control.
ModalResult	Determines whether and how the button closes its (modal) parent form.
Name	Specifies the name of the component as referenced in code.
NumGlyphs	Indicates the number of images that are in the graphic specified in the Glyph property.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Spacing	Determines where the image and text appear on a bitmap or speed button.
Style	Determines the appearance of a bitmap button.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.
WordWrap	Specifies whether the button text wraps to fit the width of the control.

4.7.1.3 Form Properties

Name	Description
Action	Designates the action associated with the control.
ActiveControl	Specifies the control that has focus on the form.
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
AlphaBlend	Specifies whether the form is translucent.
AlphaBlendValue	Specifies the degree of translucency on a translucent form.
Anchors	Specifies how the control is anchored to its parent.
AutoScroll	Indicates whether scroll bars appear automatically on the scrolling windowed control if it is not large enough to display all of its controls.
AutoSize	Specifies whether the control sizes itself

© 2015 MJT Net Ltd

	automatically to accommodate its contents.
BiDiMode	Specifies the bi-directional mode for the control.
BorderIcons	Specifies which icons appear on the title bar of the form.
BorderStyle	Specifies the appearance and behavior of the form border.
BorderWidth	Specifies the width of the control's border.
Caption	Specifies a text string that identifies the control to the user.
ClientHeight	Specifies the height (in pixels) of the form's client area.
ClientWidth	Specifies the width (in pixels) of the form's client area.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DefaultMonitor	Specifies the monitor on which the form appears.
DockSite	Specifies whether the control can be the target of drag-and-dock operations.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
FormStyle	Determines the form's style.
GlassFrame	Access the GlassFrame under Windows Vista.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpFile	Specifies the name of the file the form uses to display Help.
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
HorzScrollBar	Represents the horizontal scroll bar for the scrolling windowed control.
Icon	Specifies the icon that appears when the form is minimized. Enter the filename to load an icon onto the form. Note that the icon data is loaded statically into and stored in the dialog and while the file name is also stored for reference there is no dynamic link to that file.
KeyPreview	Specifies whether the form should receive keyboard events before the active control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.

© 2015 MJT Net Ltd

Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Menu	Specifies the form's main menu.
Name	Specifies the name of the component as referenced in code.
ObjectMenuItem	Represents an OLE object menu item that reacts to selections of OLE objects.
OldCreateOrder	Specifies when OnCreate and OnDestroy events occur.
OnTaskBar	Controls whether form should be shown on and minimized to Windows task bar. Note that in a compiled macro this will also cause the main application task bar icon to hide and remain hidden for the duration of the script.
Padding	Specifies the padding of a control.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentFont	Determines where a control looks for its font information.
PixelsPerInch	Represents the proportion of the font on the system on which the form was designed.
PopupMenu	Identifies the pop-up menu associated with the control.
PopupMode	Controls how the top-level form behaves with respect to Window's WS_POPUP style.
PopupParent	Sets an order for stacked forms that users cannot change.
Position	Represents the size and placement of the form.
PrintScale	Represents the proportions of a printed form.
Scaled	Specifies whether the form is sized according to the value of the PixelsPerInch property.
ScreenSnap	Specifies whether form snaps to edge of screen.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
SnapBuffer	Specifies distance for screen snap.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
TransparentColor	Specifies whether a color on the form appears transparent.
TransparentColorValue	Indicates the color on the form that appears transparent when TransparentColor is true.
UseDockManager	Specifies whether the docking manager is used in drag-and-dock operations.
VertScrollBar	Represents the vertical scroll bar for the scrolling windowed control.
Visible	Indicates whether the form is visible.
Width	Specifies the horizontal size of the control or form in pixels.
WindowMenu	Specifies the Window menu for an MDI parent form.
WindowState	Represents how the form appears on the screen.

© 2015 MJT Net Ltd

Note: it may not be possible to use all of these properties fully within Macro Scheduler.

4.7.1.4 MainMenu - Menu Builder

To create a menu for your dialog right click on the dialog in the Dialog Designer and select "Menu Builder".

You will be presented with a dialog showing a menu in tree form in a list. Press F2 to edit a caption. Right click and select Indent or Unindent to set the menu level (e.g. sub items are indented from the parent item). Right click to Delete or Insert items.

File
- Open
- Save As
- Exit

Edit
- Cut
- Copy
- Paste

Help
- About
- Contents

If a menu does not already exist the menu builder will show you an example tree. Overwrite it/edit it to create your own.

If a menu already exists the menu builder will show you the current menu so that you can make modifications.

To set individual properties of a menu item, once the menu has been added to the form, use the mouse to select the appropriate menu item and its properties will be displayed in the Property Grid where you can make changes. See MenuItem Object Properties 100.

Use the AddDialogHandler 106 command to add event subroutines to menu items. E.g.:

AddDialogHandler>Dialog1,MenuItem2,OnClick,DoOpenFile

4.7.1.5 Label Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
Alignment	Controls the horizontal placement of the text within the label.
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoSize	Determines whether the size of the label automatically resizes to accommodate the text.
BiDiMode	Specifies the bi-directional mode for the control.
Caption	Specifies a text string that identifies the control to the user.
Color	Specifies the background color of the control.

© 2015 MJT Net Ltd

Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
EllipsisPosition	Specifies how ellipsis is placed in text not fitting in allocated rectangle.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
FocusControl	Designates a windowed control associated with the label.
Font	Controls the attributes of text written on or in the control.
GlowSize	Specifies the radius of the glow around the label.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Layout	Specifies the vertical placement of the text within the label.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowAccelChar	Determines how an ampersand in the label text is displayed.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Transparent	Specifies whether controls that sit below the label on a form can be seen through the label.
Visible	Determines whether the component appears onscreen.

© 2015 MJT Net Ltd

Width	Specifies the horizontal size of the control or form in pixels.
WordWrap	Specifies whether the label text wraps when it is too long for the width of the label.

4.7.1.6 Edit Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
Alignment	Determines how the text is aligned within the text edit control.
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoSelect	Determines whether all the text in the edit control is automatically selected when the control gets focus.
AutoSize	Determines whether the height of the edit control automatically resizes to accommodate the text.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BevelWidth	Specifies the width of the inner and outer bevels.
BiDiMode	Specifies the bi-directional mode for the control.
BorderStyle	Determines whether the edit control has a single line border around the client area.
CharCase	Determines the case of the text within the edit control.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
HideSelection	Determines whether the visual indication of the selected text remains when focus shifts to another control.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ImeMode	Determines the behavior of the input method editor (IME).

© 2015 MJT Net Ltd

ImeName	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
MaxLength	Specifies the maximum number of characters the user can enter into the edit control.
Name	Specifies the name of the component as referenced in code.
NumbersOnly	Allows only numbers to be typed into the text edit.
OEMConvert	Determines whether characters typed in the edit control are converted from ANSI to OEM and then back to ANSI.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PasswordChar	Indicates the character, if any, to display in place of the actual characters typed in the control.
PopupMenu	Identifies the pop-up menu associated with the control.
ReadOnly	Determines whether the user can change the text of the edit control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Text	Contains a text string associated with the control.
TextHint	A hint or message to be displayed when the Text property is empty.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

Note: not all of these properties may be useable in the current implementation.

4.7.1.7 Memo Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
Alignment	Determines how the text is aligned within the text edit control.
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.

© 2015 MJT Net Ltd

BiDiMode	Specifies the bi-directional mode for the control.
BorderStyle	Determines whether the edit control has a single line border around the client area.
CharCase	Determines the case of the text within the edit control.
CharCount	The number of characters in the memo.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
HideSelection	Determines whether the visual indication of the selected text remains when focus shifts to another control.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ImeMode	Determines the behavior of the input method editor (IME).
ImeName	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
LineCount	The number of lines in the memo.
Lines	Contains the individual lines of text in the memo control. **Use the Text property to set/get the text in the memo.**
LoadFromFile	Specify a filename to load text from.
Margins	Specifies the margins for the control.
MaxLength	Specifies the maximum number of characters the user can enter into the edit control.
Name	Specifies the name of the component as referenced in code.
OEMConvert	Determines whether characters typed in the edit control are converted from ANSI to OEM and then back to ANSI.
ParentBiDiMode	Specifies whether the control uses its parent's

© 2015 MJT Net Ltd

	BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ReadOnly	Determines whether the user can change the text of the edit control.
SaveToFile	Specify a filename to save the memo text to.
ScrollBars	Determines whether the memo control has scroll bars.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Text	The text inside the memo
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
WantReturns	Determines whether the user can insert return characters into the text.
WantTabs	Determines whether the user can insert tab characters into the text.
Width	Specifies the horizontal size of the control or form in pixels.
WordWrap	Determines whether the edit control inserts soft carriage returns so text wraps at the right margin.

4.7.1.8 CheckBox Object Properties

Name	**Description**
Action	Designates the action associated with the control.
Align	Determines how the control aligns within its container (parent control).
Alignment	Determines whether the check box label aligns to the left or to the right of the tick box.
AlignWithMargins	Indicates whether a control should be constrained by margins.
AllowGrayed	Determines whether check box can be in a "grayed" state.
Anchors	Specifies how the control is anchored to its parent.

© 2015 MJT Net Ltd

BiDiMode	Specifies the bi-directional mode for the control.
Caption	Specifies a text string that identifies the control to the user.
Checked	Specifies whether the button control is checked.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
State	Indicates whether the check box is selected, deselected, or grayed.

© 2015 MJT Net Ltd

TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.
WordWrap	Specifies whether the button text wraps to fit the width of the control.

4.7.1.9 ListBox Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoComplete	Determines whether the user can give focus to items by typing in the list.
AutoCompleteDelay	Specifies the delay between a key press and an attempt to autocomplete the field.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BevelWidth	Specifies the width of the inner and outer bevels.
BiDiMode	Specifies the bi-directional mode for the control.
BorderStyle	Determines whether the list box has a border.
Color	Specifies the background color of the control.
Columns	Specifies the number of columns, in a multi-column list box, that are visible without having to scroll.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
ExtendedSelect	Makes list item multi-select operate with Shift and CTRL keys.

© 2015 MJT Net Ltd

Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ImeMode	Determines the behavior of the input method editor (IME).
ImeName	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
IntegralHeight	Determines whether the list box displays the partial items.
ItemHeight	Specifies the height in pixels of an item in an owner-draw list box.
Items	**Use the Text property to set/get the items that appear in the list box.**
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
MultiSelect	Specifies whether the user can select more than one item.
Name	Specifies the name of the component as referenced in code.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ScrollWidth	Specifies the width, in pixels, by which the list box can scroll horizontally.
SelectedIndex	Provides the index of the item selected.
SelectedItems	Returns the selected item or if MultiSelect enabled returns a %CRLF% delimited list of all selected items.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Sorted	Specifies whether the items in a list box are arranged alphabetically.
Style	Determines whether the list box is standard or owner-draw and whether it is virtual.
TabOrder	Indicates the position of the control in its parent's tab order.

© 2015 MJT Net Ltd

TabStop	Determines if the user can tab to a control.
TabWidth	Specifies the size of the tabs in the list box.
Tag	Stores an integer value as part of a component.
Text	The text of the list. Specify a list of items separated by line breaks. To enter multiple lines of text at design time right click in the property editor and select "Multiline Edit". When assigning a list of items at run time using SetDialogProperty delimit lines with the %CRLF% variable.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.10 ComboBox Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoCloseUp	Specifies whether the drop-down closes up automatically when the user selects an item.
AutoComplete	Positions to matching list items as you type.
AutoCompleteDelay	Specifies the delay between a key press and an attempt to autocomplete the field.
AutoDropDown	Specifies whether the drop-down list drops down automatically in response to user keystrokes.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BiDiMode	Specifies the bi-directional mode for the control.
CharCase	Determines the case of the text in the combo box.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
DropDownCount	Specifies the maximum number of items displayed in

© 2015 MJT Net Ltd

	the drop-down list.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ImeMode	Determines the behavior of the input method editor (IME).
ImeName	Specifies the input method editor (IME) to use for converting keyboard input to Asian language characters.
ItemHeight	Specifies the height, in pixels, of the items in the drop-down list.
ItemIndex	Specifies the index of the selected item.
Items	**Use the ListText property to get/set the item list.**
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
ListText	The text of the list. Specify a list of items separated by line breaks. To enter multiple lines of text at design time right click in the property editor and select "Multiline Edit". When assigning a list of items at run time using SetDialogProperty delimit lines with the %CRLF% variable.
Margins	Specifies the margins for the control.
MaxLength	Specifies the maximum number of characters the user can type into the edit portion of the combo box.
Name	Specifies the name of the component as referenced in code.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Sorted	Determines whether the list portion of the combo box is alphabetized.
Style	Determines the display style of the combo box.
TabOrder	Indicates the position of the control in its parent's tab order.

© 2015 MJT Net Ltd

TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Text	Contains a text string associated with the control.
TextHint	Specifies the text that is displayed as a text watermark in the edit box of the combo box control.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.11 Image Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoSize	Specifies whether the control sizes itself automatically to accommodate the dimensions of the image.
BitmapData	Can be used to set the image from binary data imported into the script. Use LabelToVar 216 to get the label data into a variable first.
Center	Indicates whether the image is centered in the image control.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
IncrementalDisplay	Specifies whether successive approximations of the image should be drawn during slow operations.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
LoadImage	Loads an image from file. Specify the filename of the picture to display (can be bitmap, jpeg, png, ico). Note that the image data is loaded statically into and

© 2015 MJT Net Ltd

	stored in the dialog. To link dynamically assign a filename to LoadImage at runtime.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
Proportional	Indicates whether the image should be changed, without distortion, so that it fits the bounds of the image control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Stretch	Indicates whether the image should be changed so that it exactly fits the bounds of the image control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Transparent	Specifies whether the background of the image obscures objects below the image object.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.12 RadioGroup Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BiDiMode	Specifies the bi-directional mode for the control.
Caption	Specifies a text string that identifies the control to the user.
Color	Specifies the background color of the control.
Columns	Specifies number of columns in the radio group.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged

© 2015 MJT Net Ltd

	normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ItemIndex	Indicates which radio button in the group is currently selected.
Items	**Use the Text property to get/set the list of radio buttons in the radio group.**
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
ParentBackground	Determines whether control uses parent's theme background.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
SelectedItem	Returns the index of the selected radio button.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Text	The text of the radio buttons. Specify a list of items separated by line breaks. To enter multiple lines of text at design time right click in the property editor and select "Multiline Edit". When assigning a list of items at run time using SetDialogProperty delimit lines with the %CRLF% variable.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.

© 2015 MJT Net Ltd

Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.
WordWrap	Specifies whether the text for the radio buttons wraps to fit the width of the radio group.

4.7.1.13 ProgressBar Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BackgroundColor	Sets and gets the color of the background of the progress bar.
BarColor	Sets and gets the color of the highlighted part of the progress bar.
BorderWidth	Specifies the width of the control's border.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
MarqueeInterval	Time in milliseconds between marquee animation updates.
Max	Specifies the upper limit of the range of possible positions.
Min	Specifies the lower limit of the range of possible positions.
Name	Specifies the name of the component as referenced in code.
Orientation	Specifies whether the progress bar is oriented

© 2015 MJT Net Ltd

	vertically or horizontally.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
Position	Specifies the current position of the progress bar.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Smooth	Specifies whether the progress bar is smooth or segmented.
SmoothReverse	Sets and gets the ability to show a decrease in progress of the progress bar.
State	Sets and gets the current state of the progress bar: pbsNormal, pbsError, or pbsPaused.
Step	Specifies the amount that Position increases when the StepIt method is called.
Style	Determines how progress is illustrated by the TProgressBar.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.14 PageControl Object Properties

Name	Description
ActivePage	Specifies the page currently displayed by the page control.
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BiDiMode	Specifies the bi-directional mode for the control.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DockSite	Specifies whether the control can be the target of drag-and-dock operations.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.

© 2015 MJT Net Ltd

DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
HotTrack	Determines whether labels on the tab under the mouse are automatically highlighted.
Images	Specifies the images drawn in tabs.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
MultiLine	Determines whether the tabs can appear on more than one row.
Name	Specifies the name of the component as referenced in code.
OwnerDraw	Specifies whether the tab control handles its own painting.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
RaggedRight	Specifies whether rows of tabs stretch to fill the width of the control.
ScrollOpposite	Determines how the rows of tabs are scrolled in a multi-line tab control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
Style	Specifies the style of the tab control.
TabHeight	Specifies the height, in pixels, of the tabs in the tab control.
TabIndex	Identifies the selected tab on a tab control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabPosition	Determines whether tabs appear at the top or bottom.
TabStop	Determines if the user can tab to a control.
TabWidth	Specifies the horizontal size, in pixels, of the tabs in the tab control.

© 2015 MJT Net Ltd

Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

To add pages to a page control right click on it in the dialog designer and select "Add Page".

4.7.1.15 TabSheet Object Properties

A TabSheet is a page of a PageControl. Add a page to a PageControl by right clicking on the PageControl and selecting "Add Page".

Name	Description
AlignWithMargins	Indicates whether a control should be constrained by margins.
BorderWidth	Specifies the width of the control's border.
Caption	Specifies a text string that identifies the control to the user.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Highlighted	Indicates whether the tab sheet appears highlighted.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
ImageIndex	Specifies an image for the tab.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
PageIndex	Indicates the index of the tab sheet in the list of tab sheets maintained by the page control.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font

© 2015 MJT Net Ltd

	information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
TabVisible	Specifies whether the tab of the TTabSheet object appears in its TPageControl.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.16 Panel Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
Alignment	Determines how the caption is aligned within the panel.
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoSize	Specifies whether the control sizes itself automatically to accommodate its contents.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Determines the style of the inner bevel of a panel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Determines the style of the outer bevel of a panel.
BevelWidth	Determines the width, in pixels, of both the inner and outer bevels of a panel.
BiDiMode	Specifies the bi-directional mode for the control.
BorderStyle	Determines the style of the line drawn around the perimeter of the panel control.
BorderWidth	Specifies the distance, in pixels, between the outer and inner bevels.
Caption	Specifies a text string that identifies the control to the user.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DockSite	Specifies whether the control can be the target of drag-and-dock operations.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory

© 2015 MJT Net Ltd

	bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
FullRepaint	Determines how the panel repaints itself when it is resized.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Locked	Determines whether a panel that is used as a toolbar is replaced by a toolbar supplied by an OLE server.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
Padding	Specifies the padding of a control.
ParentBackground	Determines whether control uses parent's theme background.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine if it should appear three dimensional.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the control.
ShowCaption	Specifies whether to display the caption of the panel control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines if the user can tab to a control.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.

© 2015 MJT Net Ltd

UseDockManager	Specifies whether the docking manager is used in drag-and-dock operations.
VerticalAlignment	Sets the vertical position of the caption.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.17 Splitter Object Properties

Name	Description
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
AutoSnap	Determines whether neighboring objects are resized to zero when the splitter is used to make them smaller than MinSize.
Beveled	Determines whether the splitter looks beveled along the edge that moves.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
MinSize	Specifies the minimum size, in pixels, of the panes on either side of the splitter.
Name	Specifies the name of the component as referenced in code.
ParentColor	Determines where a control looks for its color information.
ParentCustomHint	Determines where a control looks for its custom hint.
ResizeStyle	Specifies the effect of moving the splitter.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

© 2015 MJT Net Ltd

4.7.1.18 StatusBar Object Properties

Name	Description
Action	Designates the action associated with the control.
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Indicates whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
AutoHint	Specifies whether the status bar's text is set automatically to the current hint.
BiDiMode	Specifies the bi-directional mode for the control.
BorderWidth	Specifies the width of the control's border.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	Custom hint for control.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory bitmap first.
DragCursor	Indicates the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Determines how the control initiates drag-and-drop or drag-and-dock operations.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
Font	Controls the attributes of text written on or in the control.
Height	Specifies the vertical size of the control in pixels.
HelpContext	Numeric ID for control's context-sensitive help topic
HelpKeyword	Keyword for control's context-sensitive help topic
HelpType	Indicates whether the control's context sensitive help topic is identified by context ID or by keyword.
Hint	Contains the text string that can appear when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
Margins	Specifies the margins for the control.
Name	Specifies the name of the component as referenced in code.
Panels	Lists the panels (TStatusPanel) in the status bar.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Determines where a control looks for its color information.
ParentCustomHint	Determines where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Determines where a control looks for its font information.
ParentShowHint	Determines where a control looks to find out if its Help Hint should be shown.
PopupMenu	Identifies the pop-up menu associated with the

© 2015 MJT Net Ltd

	control.
ShowHint	Determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control.
SimplePanel	Determines whether the status bar displays a single panel or multiple panels.
SimpleText	Contains the string that is displayed in the status bar when SimplePanel is set to true.
SizeGrip	Determines whether the status bar is resizable at runtime.
Tag	Stores an integer value as part of a component.
Top	Specifies the Y coordinate of the top left corner of a control, relative to its parent or containing control in pixels.
UseSystemFont	Specifies whether the status bar uses the system font.
Visible	Determines whether the component appears onscreen.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.19 StringGrid Object Properties

Name	Description
Action	Specifies the action associated with the control.
Align	Determines how the control aligns within its container (parent control).
AlignWithMargins	Specifies whether a control should be constrained by margins.
Anchors	Specifies how the control is anchored to its parent.
BevelEdges	Specifies which edges of the control are beveled.
BevelInner	Specifies the cut of the inner bevel.
BevelKind	Specifies the control's bevel style.
BevelOuter	Specifies the cut of the outer bevel.
BevelWidth	Specifies the width of the inner and outer bevels.
BiDiMode	Specifies the bidirectional mode for the control.
BorderStyle	Determines whether a single line border is drawn around the grid.
ColCount	Specifies the number of columns in the grid.
Color	Specifies the background color of the control.
Constraints	Specifies the size constraints for the control.
Ctl3D	Determines whether a control has a three-dimensional (3-D) or two-dimensional look.
Cursor	Specifies the image used to represent the mouse pointer when it passes into the region covered by the control.
CustomHint	CustomHint is a custom hint for the control.
DefaultColWidth	Determines the width (in pixels) of all columns that have not been explicitly resized.
DefaultDrawing	Determines whether the Cells are painted when the grid is drawn.
DefaultRowHeight	Specifies the height (in pixels) of all rows that have not been explicitly resized.
DoubleBuffered	Determines whether the control's image is rendered directly to the window or painted to an in-memory

© 2015 MJT Net Ltd

	bitmap first.
DragCursor	Specifies the image used to represent the mouse pointer when the control is being dragged.
DragKind	Specifies whether the control is being dragged normally or for docking.
DragMode	Specifies how the control initiates drag-and-drop or drag-and-dock operations.
DrawingStyle	Determines the style used when drawing the grid.
Enabled	Controls whether the control responds to mouse, keyboard, and timer events.
FixedColor	Specifies the background color of the fixed rows and columns in the grid.
FixedCols	Specifies the number of columns on the left of the grid that cannot be scrolled.
FixedRows	Specifies the number of rows on the top of the grid that cannot be scrolled.
Font	Specifies the attributes of text written on or in the control.
GradientEndColor	Determines the ending gradient color.
GradientStartColor	Determines the starting gradient color.
GridLineWidth	Specifies the width (in pixels) of the lines that separate the cells of the grid.
Height	Specifies the vertical size of the control in pixels.
HelpContext	The HelpContext property contains the numeric context ID that identifies the Help topic for the control.
HelpKeyword	The HelpKeyword property contains the keyword string that identifies the Help topic for the control.
HelpType	Specifies whether the control's context-sensitive Help topic is identified by a context ID or by keyword.
Hint	Hint contains the text string that appears when the user moves the mouse over the control.
Left	Specifies the horizontal coordinate of the left edge of a component relative to its parent.
LoadFromCSV	Loads grid from CSV data
Margins	Specifies the margins of the control.
MouseInClient	Indicates whether the mouse pointer is currently in the client area of the control.
Name	Specifies the name of the component as referenced in code.
Options	Specifies various display and behavioral properties of the grid.
ParentBiDiMode	Specifies whether the control uses its parent's BiDiMode.
ParentColor	Specifies where a control looks for its color information.
ParentCtl3D	Determines where a component looks to determine whether it should have a three-dimensional look.
ParentCustomHint	Specifies where a control looks for its custom hint.
ParentDoubleBuffered	ParentDoubleBuffered defers the DoubleBuffered property of this component to the value of the parent's DoubleBuffered property.
ParentFont	Specifies where a control looks for its font information.
ParentShowHint	Specifies where a control looks to find out if its Help Hint should be shown.
PopupMenu	Specifies the pop-up menu associated with the control.

© 2015 MJT Net Ltd

RowCount	Specifies the number of rows in the grid.
SaveToCSV	Savest the grid to a CSV data string.
ScrollBars	Specifies whether the grid includes horizontal and vertical scroll bars.
ShowHint	ShowHint specifies whether to show the Help Hint when the mouse pointer moves over the control.
TabOrder	Indicates the position of the control in its parent's tab order.
TabStop	Determines whether the user can tab to a control.
TabStops	
Tag	Stores a NativeInt integral value as a part of a component.
TBDockHeight	Specifies the height of the control when it is docked vertically.
Top	Specifies the Y coordinate of the upper-left corner of a control, relative to its parent or containing control in pixels.
Touch	Touch specifies the touch manager component associated with the control.
Visible	Specifies whether the component appears onscreen.
VisibleColCount	Indicates the number of scrollable columns visible in the grid.
VisibleRowCount	Indicates the number of scrollable rows visible in the grid.
Width	Specifies the horizontal size of the control or form in pixels.

4.7.1.20 MenuItem Object Properties

Name	Description
Action	Designates the action associated with the menu item.
AutoCheck	Indicates whether the menu item's checked state toggles automatically when the item is clicked.
AutoHotkeys	Determines whether the accelerator keys for submenu items can be reset automatically.
AutoLineReduction	Determines whether redundant separator bars are automatically removed from the submenu.
BMPData	Bitmap data for data imported into macro. Use LabelToVariable to get a variable to set this property to.
Break	Determines whether the menu item starts a new column in the menu.
Caption	Specifies the text of the menu item.
Checked	Specifies whether a check mark should appear beside the Caption.
Default	Specifies whether the menu item is invoked when the parent item is double clicked.
Enabled	Specifies whether the menu item is enabled.
GroupIndex	Identifies the logical group to which the menu item belongs.
HelpContext	Specifies the help context ID associated with the menu item.
Hint	Specifies the text string that can appear when the user moves the mouse pointer over a menu item.

© 2015 MJT Net Ltd

ImageIndex	Indicates which image maintained by the parent menu appears next to the menu item.
LoadBitmap	Loads bitmap from file. Specify filename of bitmap to show against menu item. Right click in properties grid to select "File Browse" option to locate a bitmap file. Note that the image data is loaded statically into and stored in the dialog. To link dynamically assign a filename to LoadBitmap at runtime.
Name	Specifies the name of the component as referenced in code.
RadioItem	Specifies whether the menu item is mutually exclusive with other menu items in its group.
ShortCut	Specifies the key combination users can type to quickly access the menu item.
SubMenuImages	Lists the images that can appear beside individual items in a submenu of this menu item.
Tag	Stores an integer value as part of a component.
Visible	Specifies whether the menu item appears in the menu.

4.7.1.21 HTMLViewer Object Properties

Full list coming soon. In the mean time:

Use the HTML property to SET the html you want to display:

```
Let>myHTML=<H1>Hello World</H1><p>This is a paragraph</p>
SetDialogProperty>DialogName,MSHTMLViewer1,HTML,myHTML
```

4.7.2 Dialog Functions

Show 105
CloseDialog 102
GetDialogProperty 105
SetDialogProperty 106
AddDialogHandler 106
SetDialogObjectColor 103
SetDialogObjectFont 104
SetDialogObjectFocus 104

Deprecated:

GetDialogAction 103
ResetDialogAction 103
SetDialogObjectVisible 105

© 2015 MJT Net Ltd

4.7.2.1 CloseDialog

CloseDialog>DialogName

Not supported in Macro Scheduler Lite.

Closes non-modal dialog specified by DialogName

Abbreviation: CDG
See also: Show, GetDialogProperty, SetDialogProperty, AddDialogHandler

See Dialog 102 for details on creating dialogs.

4.7.2.2 Dialog

Dialog>DialogName

Not supported in Macro Scheduler Lite.

Marks the start of a custom dialog block. Since v12 it is no longer necessary to edit code within a Dialog block. Instead use the Dialog Designer 24 to create and edit dialogs.

v12 dialog block syntax is different and more advanced to that of previous versions. To preserve backward compatibility old style dialogs can still be executed and loaded into the Dialog Designer, but the Dialog Designer will only save to the new syntax.

To edit an existing dialog right click on the Dialog>DialogName line and select the Edit option from the pop up menu.

All dialogs should be created using the visual Dialog Designer 24. There is no need to edit the code inside dialog blocks produced by the Dialog Designer. Variables are not processed within dialog blocks and do not need to be since all properties can be set and retrieved using the SetDialogProperty 106 and GetDialogProperty 105 functions.

Dialogs are created when the dialog block is executed. Do not put dialog blocks inside loops or any code that could get executed more than once as you will get an error if you try to create a dialog that already exists. The dialog is created by the dialog block but is not displayed until the Show function is used. After a dialog block (so after a dialog has been created) you can get/set any properties using the GetDialogProperty 105/SetDialogProperty 106 functions and create event handlers using AddDialogHandler 106.

See also: Show 105, CloseDialog 102, GetDialogProperty 105, SetDialogProperty 106, AddDialogHandler 106

4.7.2.3 EndDialog

EndDialog>DialogName

Not supported in Macro Scheduler Lite.

© 2015 MJT Net Ltd

Marks the end of a dialog block. See Dialog 102 for details.

4.7.2.4 GetDialogAction

GetDialogAction>DialogName,Result

Not supported in Macro Scheduler Lite.

```
Deprecated.  Supported only for backward compatibility.  Works only with
old-style Dialog Blocks (v11 and earlier).
```

4.7.2.5 PrintDialog

PrintDialog>DialogName,PrintScale

Not supported in Macro Scheduler Lite.

Prints an image of the specified dialog to the default printer.

PrintScale can be one of: poNone, poProportional, poPrintToFit.

4.7.2.6 ResetDialogAction

ResetDialogAction>DialogName

Not supported in Macro Scheduler Lite.

```
Deprecated.  Supported only for backward compatibility.  Works only with
old-style Dialog Blocks (v11 and earlier).
```

4.7.2.7 SetDialogObjectColor

SetDialogObjectColor>Dialog_Name,Object_Name,Color_Code

Not supported in Macro Scheduler Lite.

Deprecated by SetDialogProperty.

Sets the background color of the specified dialog object. Sets background of dialog if Object_Name left blank.

If you are unsure of the name of the object edit the dialog in the Dialog Designer 24 and double click on the object to see it's name.

Use the RGB function to create a color code. 0 is black.

Not all types of objects can have a background color. The following objects can be set: Edits, Labels, Listboxes, Checkboxes, Memos, Dialogs.

Abbreviation: SDC

© 2015 MJT Net Ltd

Example:

```
RGB>20,50,200,aColor
SetDialogObjectColor>Dialog1,,aColor
```

4.7.2.8 SetDialogObjectFont

SetDialogObjectFont>Dialog_Name,Object_Name,Font_Name,Font_Size,Font_Style,Font_Color

Not supported in Macro Scheduler Lite.

Sets the font characteristics of the specified dialog object. Sets all objects if Object_Name left blank.

If you are unsure of the name of the object edit the dialog in the Dialog Designer 24 and double click on the object to see it's name.

Font_Style can be one of:

0: Normal
1: Bold
2: Italic
3: Underline
4: Strikeout

For a combination of font styles call SetDialogObjectFont multiple times for the same object with a different style value. E.g. for Bold and Italic call the function twice, first with Font_Style set to 1 and then the second time set to 2. To reset the font style to normal pass in zero.

Use the RGB function to create a color code. 0 is black.

You cannot change the color of button captions, as button caption colors are defined in the Windows theme settings in Control Panel.

Abbreviation: SDF

Example:

```
SetDialogObjectFont>Dialog1,msMemo2,Arial,8,1,0
```

4.7.2.9 SetDialogObjectFocus

SetDialogObjectFocus>Dialog_Name,Object_Name

Not supported in Macro Scheduler Lite.

Focuses the specified object of the specified dialog.

If you are unsure of the name of the object edit the dialog in the Dialog Designer 24 and double click on the object to see it's name.

Abbreviation: DSF

Example:

© 2015 MJT Net Ltd

```
SetDialogObjectFocus>Dialog1,msMemo2
```

4.7.2.10 SetDialogObjectVisible

SetDialogObjectVisible>Dialog_Name,Object_Name,Show(0|1)

Not supported in Macro Scheduler Lite.

Deprecated by SetDialogProperty (set Visible property to false/true).

Shows or hides the specified dialog object. Set Show to 1 to make the object visible. Set Show to 0 to hide the object.

If you are unsure of the name of the object edit the dialog in the Dialog Designer 24 and double click on the object to see it's name.

Abbreviation: SDV

Example:

```
SetDialogObjectVisible>Dialog1,msButton1,0
```

4.7.2.11 Show

Show>DialogName[,Result]

Not supported in Macro Scheduler Lite.

Displays a dialog created with a Dialog 102 block.

If a result variable is provided in Result the dialog is shown modal. Otherwise it is displayed non-modal. For non-modal dialogs see GetDialogAction 103, ResetDialogAction 103 and CloseDialog 102 for determining which button was pressed and to close the dialog.

Result is set to the modal result value of the button that was pressed. If the Dialog is closed without a button being pressed, e.g. the user pressed the close button, Result will be set to 2. Modal Result 2 means the dialog was canceled. Bear this in mind when assigning modal result values to other buttons. It is advisable to use anything other than 2. 1 is commonly used for OK.

Note that during debug mode a modal dialog is not completely modal - other dialogs are not disabled as they would be when the script is run outside of the debugger. If the dialog was shown fully modal it would not be possible to interact with the debugger.

See Dialog 102 for details on creating dialogs and an example.

4.7.2.12 GetDialogProperty

GetDialogProperty>DialogName,ObjectName,PropertyName,Result

Not supported in Macro Scheduler Lite.

Returns the value of an object's property. To access properties of the dialog itself leave ObjectName blank.

© 2015 MJT Net Ltd

To see a list of available property names edit the dialog in the Dialog Designer and select the control. The property names are displayed in the Properties grid.

See also: SetDialogProperty 106
Abbreviation: GDP

Example:

GetDialogProperty>Dialog1,Memo1,Text,strTheText

4.7.2.13 SetDialogProperty

SetDialogProperty>DialogName,ObjectName,PropertyName,NewValue

Not supported in Macro Scheduler Lite.

Sets the value of an object's property. To access properties of the dialog itself leave ObjectName blank.

To see a list of available property names edit the dialog in the Dialog Designer and select the control. The property names are displayed in the Properties grid. Where the property type is a list (i.e. can have more than one value), specify each item delimited by commas within square brackes (e.g. [biSystemMenu, biMaximize]). Look at the properties grid in the Dialog Designer to see existing property values for an example. Note that in the properties grid you are presented with a drop down box with checkboxes, indicating that more than one item can be chosen. Some properties can take one of a set number of values. In the Dialog Designer properties grid you will see all available options in a drop down box for the property concerned.

See also: GetDialogProperty 105
Abbreviation: SDP

Example:

SetDialogProperty>Dialog1,Memo1,Text,Hello World

Let>border_icons=[biSystemMenu, biMaximize]
SetDialogProperty>Dialog1,,BorderIcons,border_icons

4.7.2.14 AddDialogHandler

AddDialogHandler>DialogName,ObjectName,EventName,Subroutine

Not supported in Macro Scheduler Lite.

Assigns a subroutine to an object's event procedure. Use AddDialogHandler to make a subroutine respond to user interaction, e.g. when a button is clicked.

© 2015 MJT Net Ltd

For the dialog's own events, leave ObjectName blank.

For a list of possible event names edit the dialog in the Dialog Designer, right click on an object and select "List Event Names".

For non-modal dialogs you will almost always want to create a dialog handler for the dialog's OnClose event so that you know when the user has closed the dialog.

You can specify values to pass into the subroutine by including them within parentheses after the subroutine name. These are then set to Subroutine_Var_1, Subroutine_Var_2 etc, in the same way as parameters passed via GoSub. See examples.

See also: Dialog 102, Show 105, CloseDialog 102, GetDialogProperty 105, SetDialogProperty 106
Abbreviation: ADH

Example:

```
Dialog>Dialog1
..
End>Dialog1

AddDialogHandler>Dialog1,Button1,OnClick,DoClick(1)
AddDialogHandler>Dialog1,Button2,OnClick,DoClick(2)
AddDialogHandler>Dialog1,,OnClose,DoClose

...
...

SRT>DoClick
  MessageModal>Button clicked: %DoClick_Var_1%
End>DoClick

SRT>DoClose
  Let>Terminated=TRUE
End>DoClose
```

4.7.3 Dialog Examples

Due to the size of dialog blocks there is insufficient space in the offline manual for full code examples. Instead, please see the sample scripts that come with the software, or visit the online version of this page.

4.7.4 Modal vs Non-Modal Dialogs

Modal Windows

When a window is modal it remains active and focused until the user has finished with it and dismisses it. While it is active no other windows of the same application can be activated. A modal window is therefore normally a child window. The user needs to interact with it before control can be returned to the parent application. In effect the parent application is locked and nothing proceeds until the modal window is closed.

You'll find a good definition of Modal Windows on Wikipedia, here:

© 2015 MJT Net Ltd

http://en.wikipedia.org/wiki/Modal_window

Non-Modal Windows

So a non-modal window is the opposite. While it is active you can still activate other windows. The user can switch between windows of the same application. The window being active does not prevent the rest of the application from continuing.

Modal Dialogs

In Macro Scheduler you can create custom dialogs. These can be modal or non-modal to the script. When we refer to a modal dialog what we mean is that once the dialog is displayed, the main script code halts until the user closes the dialog. With the exception of subroutines linked to dialog events with AddDialogHandler 106, script execution is paused until the dialog is dismissed. The script cannot do any other processing while the modal dialog is active. A modal dialog is displayed with the Show command, with a return variable specified, in which the "modal result" of the dialog is reported, corresponding to which button was pressed to close the dialog. E.g.:

```
Show>MyDialog,result
If>result=10
  MessageModal>You clicked the OK button
Endif
```

In the above example a dialog was shown modally, because a result variable was specified in the Show command. A button whose modal result was set to 10 was clicked, setting the result of the Show command to that value.

In order to use modal dialogs you need to set the "ModalResult" property of any button that should close the dialog (e.g. OK and Cancel buttons), and provide a return variable in the Show function. When a button is pressed, the dialog is closed and the Show command returns the button's ModalResult value.

Note that if a button's ModalResult value is zero (the default) it will not close the dialog. To respond to button clicks for buttons that should not close the dialog, leave ModalResult for those buttons to zero and create an event handler with the AddDialogHandler 106 command.

Most dialogs would be modal, as in most cases you will want to pause execution of the script while the dialog is visible.

A note for existing users: If you are coming from Macro Scheduler prior to version 12 you will find you no longer need to make your dialog non-modal in order to interact with it and modify its properties, as all event handlers are now executed instantly, whether or not the dialog is modal. Prior to v12 the most common reason for making a dialog non-modal was so that you could detect changes on it and update it due to the fact that scripts were executed sequentially. AddDialogHandler 106 creates real time event code that can be executed at the same time as other code and even when the dialog is modal. Therefore in most cases you should find you no longer need non-modal dialogs.

So, to recap, when using *Show>dialogname,result* the code after the Show command does not get executed until the dialog has been closed. The exception to this is any event handler code in subroutines used by AddDialogHandler 106 for the dialog concerned.

Non-Modal Dialogs

A dialog can be made to be non-modal. In which case the script carries on even after it has displayed the dialog. The user can interact with the dialog while the script continues to perform other tasks.

© 2015 MJT Net Ltd

To display a dialog non-modally specify the Show function without a result variable. E.g.:

Show>MyDialog

Since a non-modal dialog does not halt execution of the script, your script will usually need to be doing something - if there is no code after the Show command, your script will end and you will barely have time to see the dialog appear and then disappear before blinking! Any code after the Show command will be executed immediately - as soon as the dialog is made visible.

A non-modal dialog would be useful to display progress during a long or looping script, or where you have more than one dialog that need to interact with each other.

4.8 Excel Functions

4.8.1 XLAddSheet

XLAddSheet>XLBookHandle,SheetName

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Adds a worksheet to the workbook referenced by XLBookHandle. The value in XLBookHandle is returned by a call to XLCreate or XLOpen which return a reference to a workbook.

Abbreviation: XAS
See also: XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLCreate>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLAddSheet>xlBook,Customers
XLSetCell>xlBook,Customers,1,1,CustID,xlRes
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.2 XLCreate

XLCreate>Filename,Visible,XLBookHandle

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Creates a new Excel workbook with the given filename. Set Visible to 1 to make the workbook visible, or zero to hide it. Returns a handle in XLBookHandle which should be used to reference the workbook in other XL commands.

© 2015 MJT Net Ltd

Abbreviation: XLC
See also: XLAddSheet 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLCreate>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLAddSheet>xlBook,Customers
XLSetCell>xlBook,Customers,1,1,CustID,xlRes
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.3 XLDelCol

XLDelCol>XLBookHandle,SheetName,Col

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Deletes the specified column in the given workbook and sheet name. Requires a column range name. E.g. "A:A" for column A.

XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XDC
See also: XLAddSheet 109, XLCreate 109, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLDelCol>xlBook,Sheet1,D:D
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.4 XLDelRow

XLDelRow>XLBookHandle,SheetName,Row

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Deletes the specified row in the given workbook and sheet name. XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XDR
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

© 2015 MJT Net Ltd

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLDelRow>xlBook,Sheet1,3
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.5 XLDelSheet

XLDelSheet>XLBookHandle,SheetName

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Deletes the specified sheet in the workbook referenced by XLBookHandle. XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XDS
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLDelSheet>xlBook,Sheet2
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.6 XLGetCell

XLGetCell>XLBookHandle,SheetName,Row,Col,Result

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Retrieves the value in cell of given row,col coordinates from specified sheet name in workbook referenced by XLBookHandle. XLBookHandle is a handle returned by XLCreate/XLOpen.

Result is a variable name to store the retrieved value in.

Abbreviation: XGC
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLGetCell>xlBook,Sheet1,2,2,strValue
```

© 2015 MJT Net Ltd

4.8.7 XLGetSheetDims

XLGetSheetDims>XLBookHandle,SheetName,MaxRows,MaxCols

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Returns the number of rows and columns containing data in the given sheet. XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XSD
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLGetSheetDims>xlBook,Customers,rows,cols
```

4.8.8 XLOpen

XLOpen>Filename,Visible,XLBookHandle[,password]

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Opens an existing Excel workbook with the given filename. Set Visible to 1 to make the workbook visible, or zero to hide it. Returns a handle in XLBookHandle which should be used to reference the workbook in other XL commands.

If the workbook is password protected the password can be supplied in the password parameter to prevent the user being prompted.

Abbreviation: XLO
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLQuit 113, XLRun 113, XLSave 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLAddSheet>xlBook,Customers
XLSetCell>xlBook,Customers,1,1,CustID,xlRes
```

© 2015 MJT Net Ltd

4.8.9 XLQuit

XLQuit>XLBookHandle

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Closes the workbook referenced by XLBookHandle. XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XLQ
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLRun 113, XLSave 113, XLSetCell 114

Example

XLQuit>xlBook

4.8.10 XLRun

XLRun>XLBookHandle,MacroName

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Runs the specified macro in the Excel workbook referenced by XLBookHandle. XLBookHandle is a handle returned by XLCreate/XLOpen.

Abbreviation: XLR
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLSave 113, XLSetCell 114

Example

XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLRun>xlBook,CalculateTotals

4.8.11 XLSave

XLSave>XLBookHandle,FileName[,fileformat]

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Saves the specified workbook to the given filename. XLBookHandle is a handle returned by XLCreate/XLOpen.

To save to a different format (.e.g. to CSV or a different version of Excel) set the optional fileformat

© 2015 MJT Net Ltd

value to an xlFileFormat enumerator.

Abbreviation: XLS
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSetCell 114

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLAddSheet>xlBook,Customers
XLSetCell>xlBook,Customers,1,1,CustID,xlRes
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.12 XLSetCell

XLSetCell>XLBookHandle,SheetName,Row,Col,Value,Result

Not supported in Macro Scheduler Lite.

This function requires Microsoft Excel to be installed.

Sets the value in cell of given row,col coordinates from specified sheet name in workbook referenced by XLBookHandle. XLBookHandle is a handle returned by XLCreate/XLOpen.

Result will contain zero if the operation was successful, otherwise will contain the Excel error message.

Value is the value to set.

Abbreviation: XSC
See also: XLAddSheet 109, XLCreate 109, XLDelCol 110, XLDelRow 110, XLDelSheet 111, XLGetCell 111, XLGetSheetDims 112, XLOpen 112, XLQuit 113, XLRun 113, XLSave 113

Example

```
XLOpen>%USERDOCUMENTS_DIR%\mybook.xls,1,xlBook
XLSetCell>xlBook,Sheet1,2,2,'205,res
XLSave>xlBook,%USERDOCUMENTS_DIR%\mybook.xls
XLQuit>xlBook
```

4.8.13 XLSheetToArray

XLSheetToArray>file_name,sheet_name,array_name

Not supported in Macro Scheduler Lite.

Imports a sheet from a .xls Excel file (.xls only) into an array. Note that this function differs from the other XL functions in that it does NOT require Excel to be installed and instead reads the Excel file directly. Therefore the same degree of compatibility cannot be guaranteed. If you are using modern

© 2015 MJT Net Ltd

.xlsx files, export to .xls first.

The array takes the format:

arrayname_row_col
arrayname_row_count
arrayname_count

E.g. the following sheet will produce the array variables given below:

Excel Sheet "Sheet1"

ID,Date,Value
1,28/02/2010,50
2,01/03/2010,75

Code:

XLSheetToArray>file.xlsx,Sheet1,xlData

Array:

xlData_1_count=3
xlData_1_1=ID
xlData_1_2=Date
xlData_1_3=Value
xlData_2_count=3
xlData_2_1=1
xlData_2_2=28/02/2010
xlData_2_3=50
xlData_3_count=3
xlData_3_1=2
xlData_3_2=01/03/2010
xlData_3_3=75
xlData_count=3

4.9 File Handling

4.9.1 AppendFile

AppendFile>sourcefile1,sourcefile2,newfile,result

Appends sourcefile1 and sourcefile2 into newfile. result is the number of bytes written.

Abbreviation : App
See also: MoveFile 126, DeleteFile 119, IfFileExists 124, CopyFile 116

Example

```
AppendFile>c:\temp\file1.txt,c:\temp\file2.txt,c:\temp\newfile.txt,result
```

© 2015 MJT Net Ltd

4.9.2 ChangeDirectory

ChangeDirectory>path

Changes the current directory to the directory specified in path. path can be a literal string or a variable.

Abbreviation : Cha
See also: CreateDir 118

Example

```
Change Directory>c:\program files\my directory\
```

4.9.3 CopyFile

CopyFile>sourcepath,destinationpath

Copies the file (or files) or folder, sourcepath, to destinationpath

sourcepath, and destinationpath may be variables. Wildcards can be used. Source and destination may be folders.

By default if destinationpath already exists it will not be overwritten and the new file will be renamed appropriately. It is possible to change the behaviour of the CopyFile command to overwrite instead by setting CF_OVERWRITE to 1. The default value of CF_OVERWRITE is 0.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Cop
See also: MoveFile 126, DeleteFile 119, IfFileExists 124, AppendFile 115, RenameFile 128

Example

```
CopyFile>c:\temp\myfile.txt,c:\my documents\myfile.old
```

Or with variables:

```
Let>filename=c:\temp\myfile.txt
Let>newfilename=c:\temp\myfile.new
CopyFile>filename,newfilename
```

4.9.4 CopyFolder

CopyFolder>sourcepath,destinationpath

Copies the folder, sourcepath, to destinationpath

© 2015 MJT Net Ltd

sourcepath, and destinationpath may be variables. Wildcards can be used. Source and destination may be folders.

By default if destinationpath already exists it will not be overwritten and the new folder will be renamed appropriately. It is possible to change the behaviour of the CopyFolder command to overwrite instead by setting CF_OVERWRITE to 1. The default value of CF_OVERWRITE is 0.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Cop
See also: MoveFile 126, DeleteFile 119, IfFileExists 124, AppendFile 115, RenameFile 128

Example

```
CopyFolder>c:\old,c:\new
```

Or with variables:

```
Let>filename=c:\old
Let>newfilename=c:\new
CopyFolder>filename,newfilename
```

4.9.5 CountDirs

CountFiles>file_spec,result,subdir_flag

Returns the number of directories matching file_spec in the result variable.

file_spec could be a directory path e.g. c:\temp\

To recurse sub directories set subdir_flag to 1.

Abbreviation : Co2
See Also: CountFiles 118

Examples

```
CountDirs>c:\my documents\,DirCount,0
Message>Number of folders : %DirCount%
```

The following example will count all the folders anywhere on the c: drive - be warned - this can take a long time:

```
CountFiles>c:\,TotalFolders,1
Message>Total Number Of Dirs On PC : %TotalFolders%
```

© 2015 MJT Net Ltd

4.9.6 CountFiles

CountFiles>file_spec,result,subdir_flag

Returns the number of files matching file_spec in the result variable.

file_spec can include a directory path, and should include a mask. e.g. c:\temp*.*

To recurse sub directories set subdir_flag to 1.

Abbreviation : Cou
See Also: CountDirs 117

Examples

```
CountFiles>c:\my documents\*.doc,DocCount,0
Message>Number of .doc files : %DocCount%
```

The following example will count all the files anywhere on the c: drive.

```
CountFiles>c:\*.*,TotalFiles,1
Message>Total Number Of Files On PC : %TotalFiles%
```

4.9.7 CreateDir

CreateDir>directory

Creates a new directory. The directory may be a full path.

Abbreviation : Cre
See also: Change Directory 116

Example

```
CreateDir>c:\my documents\test
```

or

```
Let>dir=c:\my documents\test
CreateDir>dir
```

4.9.8 CSVFileToArray

CSVFileToArray>CSVFile,ArrayVariable

Reads the given CSV file into a multi-dimensional array.

The array takes the format:
arrayname_row_col
arrayname_row_count
arrayname_count

E.g. the following CSV data will produce the array variables given below:

© 2015 MJT Net Ltd

CSV Data:
ID,Date,Value
1,28/02/2010,50
2,01/03/2010,75

Code:
CSVFileToArray>c:\myfile.csv,csvData

Array:
csvData_1_count=3
csvData_1_0=ID
csvData_1_1=Date
csvData_1_2=Value
csvData_2_count=3
csvData_2_0=1
csvData_2_1=28/02/2010
csvData_2_2=50
csvData_3_count=3
csvData_3_0=2
csvData_3_1=01/03/2010
csvData_3_2=75
csvData_count=3

Note: The column count starts at 0. i.e. column A is 0, and not 1.

Thus :
csvData_1_0 returns the contents of Row1 Col1
csvData_1_1 returns the contents of Row1 Col2

Note that it is also possible to read CSV file data into an array using DBQuery 61 with a suitable CSV file connection string, but CSVFileToArray is less format sensitive and can cope with rows with different field counts. There are pros and cons to both methods. DBQuery can retrieve the data more selectively whereas CSVFileToArray reads in the entire file.

Example

```
CSVFileToArray>c:\temp\test.txt,csvData
If>csvData_count>0
  Let>row=0
  Repeat>row
    Let>row=row+1
    Let>field1=csvData_%row%_0
    Let>field2=csvData_%row%_1
    Let>field3=csvData_%row%_2
    MessageModal>%field1% %field2% %field3%
  Until>row=csvData_count
Endif
```

4.9.9 DeleteFile

DeleteFile>path

Deletes the file/files/folder in path. Can also be used to delete folders.

© 2015 MJT Net Ltd

path can be a variable. Wildcards can be used.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Del
See also: CopyFile 116, MoveFile 126, IfFileExists 124

Example

```
DeleteFile>c:\temp\*.*
```

4.9.10 DeleteFolder

DeleteFolder>path

Deletes the file/files/folder in path.

path can be a variable. Wildcards can be used.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Del
See also: CopyFile 116, MoveFile 126, IfFileExists 124

Example

```
DeleteFolder>c:\temp
```

4.9.11 EditIniFile

EditIniFile>inifile,section,entry,newvalue

Edits a section entry in an ini file. inifile can be a full path.

All parameters can be variables containing strings.

Abbreviation : Edi
See also: ReadIniFile 127

Example

```
EditIniFile>c:\program files\myini.ini,settings,user,fred
```

© 2015 MJT Net Ltd

4.9.12 FileDate

FileDate>filename,result

Returns the file date of filename in the result variable.

The date returned is in the format YYYYMMDD

filename can include a full path.

Abbreviation : FDT
See also: FileTime 121, FileSize 121

Example

```
FileDate>c:\program files\myfile.exe,MyFileDate
```

4.9.13 FileSize

FileSize>filename,result

Returns the size of filename in the result variable.

The size is returned in bytes.

filename can include a full path.

Abbreviation : FSZ
See also: FileDate 121, FileTime 121

Example

```
FileSize>c:\program files\myfile.exe,MyFileSize
```

4.9.14 FileTime

FileTime>filename,result

Returns the file time of filename in the result variable.

The time returned is in the format HHMMSS

filename can include a full path.

Abbreviation : FTM
See also: FileDate 121, FileSize 121

Example

```
FileTime>c:\program files\myfile.exe,MyFileTime
```

© 2015 MJT Net Ltd

4.9.15 GetFileList

GetFileList>filespec,result[,delimiter]

GetFileList returns a list of the files found matching the specified filespec. For instance, to return the list of files in the temp directory specify c:\temp*.* as the filespec.

If delimiter is ommitted each filename is separated by a semicolon (;). Otherwise the filenames are separated by the delimiter specified.

By default GetFileList returns files. To return directories only set GFL_TYPE to 1 (Let>GFL_TYPE=1).

For finer control of what kind of files get returned use GFL_ATTRIBS by setting it to one or more of the following values:

ReadOnly = 1
Hidden = 2
SysFile = 4
Directory = 16
Archive = 32
Normal = 128
SymLink = 1024
Compressed = 2048
Encrypted = 16384
Virtual = 65536
AnyFile = 511 (Default - same as without using GFL_ATTRIBS)

For multiple options add values together. E.g. to retrieve only hidden and readonly files specify a value of 3. For just read only files specify 1.

Abbreviation : GFL

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
```

4.9.16 GetNewestFile

GetNewestFile>filespec,result

Returns the newest file matching the given filespec.

See also: GetOldestFile

Example:

© 2015 MJT Net Ltd

```
//gets the newest file in the My Documents folder
GetNewestFile>%USERDOCUMENTS_DIR%\*.*,newFile

//gets the newest text file in c:\folder
GetNewestFile>c:\folder\*.txt,newFile
```

4.9.17 GetOldestFile

GetOldestFile>filespec,result

Returns the oldest file matching the given filespec.

See also: GetNewestFile 122

Example:

```
//gets the oldest file in the My Documents folder
GetOldestFile>%USERDOCUMENTS_DIR%\*.*,oldFile

//gets the oldest text file in c:\folder
GetOldestFile>c:\folder\*.txt,oldFile
```

4.9.18 IfDirExists

IfDirExists>directory_name[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If directory_name exists the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the directory specified in directory_name exists. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

Abbreviation : IFD
See also: Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfFileExists 124, Subroutines 197

Example

```
IfDirExists>c:\myfolder\temp
  MessageModal>temp folder exists - good
Else
  CreateDir>c:\myfolder\temp
Endif
```

© 2015 MJT Net Ltd

4.9.19 IfFileChanged

IfFileChanged>filename,range[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If the given file's date is in the range of days specified the first statements are executed, otherwise the else statements are executed. Alternatively, instead of Else/Endif specify label names.

When label names are specified execution jumps to the specified label if the given file's date is in the range of days specified. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

Abbreviation : IFC
See also: Label 202, Goto 200, IfWindowOpen 151, IfFileExists 124, If 146, Subroutines 197

Example

To see if test.txt is less than 30 days old :

```
IfFileChanged>test.txt,<30
  MessageModal>test.txt is less than 30 days old
Endif
```

4.9.20 IfFileExists

IfFileExists>filename[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If filename exists the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the file specified in filename exists. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

Abbreviation : IFE
See also: IfNotFileExists 150, Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfDirExists 123, Subroutines 197

Example

```
IfFileExists>c:\myfolder\myfile.txt
  ReadFile>c:\myfolder\myfile.txt,fileContents
Endif
```

© 2015 MJT Net Ltd

4.9.21 IfNotDirExists

IfNotDirExists>directory_name[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If directory_name does not exist the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the directory specified in directory_name does not exist. If a second false label is specified, execution will jump to that label if the expression resolves false (directory does exist). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

See also: IfDirExists 123, Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfFileExists 124, Subroutines 197

Example

```
IfNotDirExists>c:\myfolder\temp
  CreateDir>c:\myfolder\temp
Endif
```

4.9.22 IfNotFileChanged

IfNotFileChanged>filename,range[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If the given file's date is not in the range of days specified the first statements are executed, otherwise the else statements are executed. Alternatively, instead of Else/Endif specify label names.

When label names are specified execution jumps to the specified label if the given file's date is not in the range of days specified. If a second false label is specified, execution will jump to that label if the expression resolves false (file date IS in range). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

See also: IfFileChanged 124, Label 202, Goto 200, IfWindowOpen 151, IfFileExists 124, If 146, Subroutines 197

Example

To see if test.txt is not less than 30 days old :

```
IfNotFileChanged>test.txt,<30
  MessageModal>file must be 30 or more days old
```

© 2015 MJT Net Ltd

```
Endif
```

4.9.23 IfNotFileExists

IfNotFileExists>filename[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If filename does not exist the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the file specified in filename does not exist. If a second false label is specified, execution will jump to that label if the expression resolves false (the file exists). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

See also: IfFileExists 124, Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfDirExists 123, Subroutines 197

Example

```
IfNotFileExists>c:\myfolder\myfile.txt
  WriteLn>c:\myfolder\myfile.txt,wres,This is a New File
Endif
```

4.9.24 MoveFile

MoveFile>sourcepath,destinationpath

Moves the file (or files), sourcepath, to the file (or files) destinationpath.

sourcepath, and destinationpath may be variables. Wildcards can be used.

By default if destinationpath already exists it will not be overwritten and the new file will be renamed appropriately. It is possible to change the behaviour to overwrite instead by setting CF_OVERWRITE to 1. The default value of CF_OVERWRITE is 0.

To make MoveFile perform a simple Rename function, rather than physically moving the entire file, set MF_RENAME to 1.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Mov
See also: CopyFile 116, DeleteFile 119, IfFileExists 124, AppendFile 115, RenameFile 128

© 2015 MJT Net Ltd

Example

```
MoveFile>c:\temp\myfile.txt,c:\temp\myfile.bak
```

Or with variables:

```
Let>filename=c:\temp\myfile.txt
Let>newfilename=c:\temp\myfile.bak
MoveFile>filename,newfilename
```

4.9.25 ReadFile

ReadFile>file_name,result

Reads the entire file contents into the specified result variable. If the file does not exist, result is set to ##NOFILE##.

Abbreviation : RFL
See also: ReadLn 127, WriteLn 128

Example

```
ReadFile>c:\temp\test.txt,file
MessageModal>file
```

4.9.26 ReadIniFile

ReadIniFile>inifile,section,entry,result

Reads a value from an entry in an ini file and places the result in a variable.

All parameters can be variables containing strings.

Abbreviation : Rea
See also: EditIniFile 120

Example

This example reads a username from an ini file and displays it in a message :

```
ReadIniFile>c:\program files\myini.ini,settings,user,username
Let>msg=The Username is
ConCat>msg,username
Message>msg
```

4.9.27 ReadLn

ReadLn>file_name,line_number,result

Reads the specified line from the given text file. If line_number is past the end of the file, result will contain ##EOF##. If the file does not exist, result is set to ##NOFILE##. If an error occurred, result is set to ##ERR## followed by the error code.

Abbreviation : RLN

© 2015 MJT Net Ltd

See also: ReadFile 127, WriteLn 128

Example

This example reads each line from the text file and displays it in a Message window.

```
Let>k=1
While>line<>##EOF##
  ReadLn>c:\temp\test.txt,k,line
  If>line<>##EOF##
    MessageModal>line
  Endif
  Let>k=k+1
EndWhile
```

4.9.28 RenameFile

RenameFile>file,newfile

Renames file to newfile.

By default if newfile already exists it will not be overwritten and the new file will be renamed appropriately. It is possible to change the behaviour to overwrite instead by setting CF_OVERWRITE to 1. The default value of CF_OVERWRITE is 0.

This function does the same as MoveFile 126 with MF_RENAME set to 1.

The outcome of the operation will be reported in variables CF_RESULT and CF_RESULT_CODE. If the operation was aborted or could not proceed CF_RESULT will be False. CF_RESULT_CODE will contain a numeric code returned by the operating system where 0 indicates success. For details of the codes that may be returned see the SHFileOperation function documentation at msdn.microsoft.com.

To enable operating system file operation animations set CF_ANIMATE to 1.

Abbreviation : Ren
See also: CopyFile 116, DeleteFile 119, IfFileExists 124, AppendFile 115, MoveFile 126

Example

```
RenameFile>c:\temp\myfile.txt,c:\temp\myfile.bak
```

4.9.29 WriteLn

WriteLn>file_name,result,line

Writes text to a text file. If the file does not exist it is created and the given line is written to it. If it does exist, the line is written to the end of the file. If the command was successful, result is set to zero. A non zero result means an error occurred.

By default the text file is created with ANSI encoding. To create a Unicode file set WLN_ENCODING to UNICODE. For UTF8 set WLN_ENCODING to UTF8. Any other value sets the encoding to ANSI.

© 2015 MJT Net Ltd

WriteLn will then output Unicode. If the file already exists WriteLn will detect the file's encoding and use that encoding when writing out the data.

To omit line break characters from the end of the line set WLN_NOCRLF to 1 before issuing the WriteLn command. This will prevent subsequent text written with WriteLn appearing on a new line.

Abbreviation : WLN
See also: ReadLn 127, ReadFile 127

Example

```
GetDate>date
WriteLn>c:\temp\test.txt,result,%date% - Starting Macro
```

4.9.30 ZipAddFiles

ZipAddFiles>archive,filespec,compression,folders,result

Adds the specified files in filespec to the specified zip archive. If the archived does not already exist it is created.

Compression can be set to any number between 0 and 9 where 9 is best compression.

Set folders to 1 to include folders in the zip file, 0 to include only files.

result will return 0 if successful, 1 if an error occurred.

Abbreviation: ZAF
See also: ZipExtractFiles 129

Example

```
ZipAddFiles>c:\new\my.zip,c:\images\*.bmp,9,0,zipres
```

4.9.31 ZipExtractFiles

ZipExtractFiles>archive,filespec,destination_folder,result

Extracts from the specified zip archive the file or files in filespec to the given destination folder.

If successful the result variable will be set to 0, otherwise if an error code it will be 1.

Abbreviation: ZEF
See also: ZipAddFiles 129

Example

```
ZipExtractFiles>c:\new\my.zip,image_1.bmp,c:\image_files,zipres
```

© 2015 MJT Net Ltd

4.10 FTP/HTTP/Telnet/Email Commands

4.10.1 FTPDelFile

FTPDelFile>Server,Username,Password,port,Host_File_Spec

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and delete the file spec specified.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : FDF
See also FTPGetFile 132, FTPPutFile 134, FTPRenameFile 136, FTPGetDirList 131

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Example

```
FTPDelFile>ftp.domain.com,anonymous,user@domain.com,21,/pub/readme.txt
```

© 2015 MJT Net Ltd

4.10.2 FTPGetDirList

FTPGetDirList>Server,Username,Password,port,Local_File,Host_Dir,Host_File_Spec,Type

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and download a directory listing for the file spec specified.

The directory listing is output to the local text file given in Local_File.

Type can be either D or L, where D obtains a full directory listing, including directories, subdirectories and their contents. L creates a simple file list, containing just filenames matching the file spec.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : FGD
See also FTPGetFile 132, FTPPutFile 134, FTPDelFile 130, FTPRenameFile 136

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Example

The following command will retrieve a listing of the entire contents of the server.

© 2015 MJT Net Ltd

```
FTPGetDirList>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\readme.txt,/,*,D
```

4.10.3 FTPGetFile

FTPGetFile>Server,Username,Password,port,Local_Path,Host_Dir,Host_Filespec,Mode

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and retrieve a file or files.

The file(s) retrieved from the server is specified using Host_Dir and Host_Filespec. The file(s) is saved on the local machine under the name specified in Local_Path. If Host_Filespec is a file spec which matches multiple files (includes wildcards), all matching files will be downloaded and stored to Local_Path. Note that Local_Path can be a path to a directory or file. Specify a directory when retrieving multiple files. A filename is only sensible when retrieving a single file where the local filename should be different. Depending on the server type Host_Dir can be left empty if retrieving files from the root folder.

Mode can be either A or I, where A represents ASCII transfer mode and I represents binary.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

If passive mode is required, set FTP_PASSIVE to 1. Set to zero to switch passive mode off.

The result of the FTP operation is stored in FTP_RESULT. The number of files transferred is stored in FTP_NUMFILES.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

© 2015 MJT Net Ltd

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Abbreviation : FGF
See also: FTPPutFile 134, FTPGetDirList 131, FTPDelFile 130, FTPRenameFile 136

Example

```
FTPGetFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\myfile.txt,/pub/,readme.txt,A

Multiple Files:

FTPGetFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\,/pub/,*.txt,A
```

4.10.4 FTPMakeDir

FTPMakeDir>Server,Username,Password,port,Host_Folder_Path

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and create the folder specified.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : FMD
See also FTPGetFile 132, FTPPutFile 134, FTPRenameFile 136, FTPGetDirList 131

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:

© 2015 MJT Net Ltd

http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Example

```
FTPMakeDir>ftp.domain.com,user1,password,21,/user1/myfiles/
```

4.10.5 FTPPutFile

FTPPutFile>Server,Username,Password,port,Local_Filespec,Host_Dir,Host_File,Mode

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and upload a file or files.

The file to be uploaded is specified in Local_Filespec, which can contain a full path and wildcards. Depending on the server type Host_Dir can be left empty if adding files to the root folder. When uploading multiple files leave Host_File empty, or leave Host_File empty in order to use the same file name as the local file. You would only set Host_File when uploading one file and you want to store it with a different name.

Mode can be either A or I, where A represents ASCII transfer mode and I represents binary.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

If passive mode is required, set FTP_PASSIVE to 1. Set to zero to switch passive mode off.

The result of the FTP operation is stored in FTP_RESULT. The number of files transferred is stored in FTP_NUMFILES.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the

© 2015 MJT Net Ltd

FTP_TIMEOUT variable.

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Abbreviation : FPF
See also FTPGetFile 132, FTPGetDirList 131, FTPDelFile 130, FTPRenameFile 136

Example

```
FTPPutFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\readme.txt,/pub/,readme.txt,A

Multiple Files:

FTPPutFile>ftp.domain.com,anonymous,user@domain.com,21,c:\temp\*.txt,/pub/,,A
```

4.10.6 FTPRemoveDir

FTPRemoveDir>Server,Username,Password,port,Host_Folder_Path

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and delete the folder specified.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

© 2015 MJT Net Ltd

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

Abbreviation : FDF
See also FTPGetFile 132, FTPPutFile 134, FTPRenameFile 136, FTPGetDirList 131

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Example

```
FTPRemoveDir>ftp.domain.com,user1,password,21,/user1/myfiles/
```

4.10.7 FTPRenameFile

FTPRenameFile>Server,Username,Password,port,Host_File_Spec,New_File_Name

Not supported in Macro Scheduler Lite.

This command will connect to the specified FTP server and rename the host file specified with New_File_Name.

When the FTP commands are active a small floating window is displayed at the top left of the screen showing the status of the FTP session. To stop the status window appearing set FTP_STATUS to 0 before issuing an FTP command. Set it back to 1 to show the window. The default value for FTP_STATUS is 1.

The result of the FTP operation is stored in FTP_RESULT.

If you are connecting via a proxy server, enter the name or IP address of the proxy server in 'Server', and for 'Username' specify username@remote_server.

By default the timeout for the FTP commands is set to 15 seconds. To change this set the FTP_TIMEOUT variable.

© 2015 MJT Net Ltd

Abbreviation : FRF
See also FTPGetFile 132, FTPPutFile 134, FTPDelFile 130, FTPGetDirList 131

TLS/SSL Support

To use TLS/SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

For TLS/SSL support set FTP_USETLS to one of the following values:

0	No TLS/SSL Support	Default - no encryption
1	Use Implicit TLS	You wish to use implicit TLS. In implicit TLS FTP, you connect to port 990, start TLS negotiation, and the entire session is encrypted. IETF has depreciated implicit TLS FTP so this setting should only be used with FTP servers that do not yet support explicit TLS.
2	Use Require TLS	You wish your session to use explicit TLS. In explicit TLS FTP, you connect to port 21 like the unencrypted version of FTP and encryption lasts until you disconnect. No attempt is made to continue the FTP session if TLS negotiation fails.
3	Use Explicit TLS	You wish your session to use explicit TLS if the FTP server supports it.

If needed you can also specify the SSL certificate and key files to use using SSL_CERT, SSL_ROOT_CERT, SSL_KEY

Example

```
FTPRenameFile>ftp.domain.com,anonymous,user@domain.com,21,/pub/readme.txt,/pub/readme.new
```

4.10.8 HTTPRequest

HTTPRequest>URL,[LocalFilename],Method,
[POST_Data],Result_Variable[,ProxyServer,ProxyPort,ProxyUsername,ProxyPassword]

Not supported in Macro Scheduler Lite.

Retrieves a web document via the HTTP protocol using either GET or POST methods.

URL: URL of document to retrieve
LocalFileName: Optional (may be left blank) - local file to save response to.
Method: GET, POST, PUT or DELETE
Post_Data: Data to Post to URL if using POST method. Use name=value pairs separated by '&'. See example below.
Result_Variable: Stores result of operation. If LocalFileName is not specified and the operation is successful this will contain the HTML returned. If LocalFileName was specified and the data was successfully written to the file Result_Variable will be blank. Otherwise it will contain an error message.
ProxyServer: Optional - if using a proxy server set this to domain or IP address of proxy server.
ProxyPort: Optional - if using a proxy server set to port number of proxy server.
ProxyUsername: Optional - if using a proxy server that needs a username.
ProxyPassword: Optional - if using a proxy server that needs a password.

To post files use the HTTP_POSTFILES variable. HTTP_POSTFILES takes name=value pairs separated by '&' in the same format as Post_Data but for 'file' parameters. Post_Data can be empty

© 2015 MJT Net Ltd

if only files are being posted.

By default there is no timeout for the HTTPRequest command and requests will wait indefinitely if the server fails to respond. To change this set the HTTP_TIMEOUT value to the number of seconds to wait.

By default HTTPRequest will automatically resolve redirects. To disable this behaviour set HTTP_REDIRECTS to 0.

For basic authentication where a username and password is required by the server before the request will complete put the username and password in the URL using the following format: http://username:password@www.server.com/etc.etc

For SSL (https) connections set HTTP_SSL to 1. To use SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html

Other options:
- Set the HTTP_CHARSET variable to a character encoding name to override the default character set
- Set the HTTP_USERAGENT variable to override the user agent string that HTTPRequest sends

Abbreviation : HTT

Example

The following line does a simple GET request and saves the resulting HTML to a variable called HTMLResponse:

```
HTTPRequest>http://www.mjtnet.com,,GET,,HTMLResponse
```

The following line does the same thing but saves the output to a file instead of a variable:

```
HTTPRequest>http://www.mjtnet.com,d:\HTML\mjtnet.html,GET,,HTMLResponse
```

This demonstrates a POST operation, sending name=value pairs to the page:

```
Let>PostData=email=myemail@home.com&name=Joe Bloggs
HTTPRequest>http://www.someplace.com/someform.html,,POST,PostData,HTMLResponse
```

Posting files:

```
Let>HTTP_POSTFILES=upload1=c:\stuff\logo.gif
HTTPRequest>http://www.someplace.com/form.html,,POST,,HTMLResponse
```

4.10.9 IsConnectedToInternet

IsConnectedToInternet>result

Returns True if the a connection to the internet appears to exist. False if not connected.

Abbreviation: ISC

© 2015 MJT Net Ltd

4.10.10 IEGetTags

IEGetTags>URL,Tagname,ReturnType,ArrayVar

This function will return an array of tag values for all tags of the given tag name in the IE window whose URL matches or contains the specified URL.

URL: The current URL of the IE window to extract from. This can be a substring. If using a substring and you have multiple IE windows open try to be unique to ensure a correct match.

Tagname: The name of the tags to extract. E.g. "A", "INPUT" or "DIV". This can be any tag type.

ReturnType: Use: "H" to return innerHTML (i.e. the HTML between the opening and closing tags); "O" to return outerHTML (includes the HTML in the tag itself) or "T" for innerText (the text that appears between the opening and closing tags).

ArrayVar: The variable name to store the results in. ArrayVar_Count will contain the number of items returned.

This function can be used to extract data from web pages. Use a loop and RegEx/EasyPatterns or similar to further identify specific elements.

Fore more control of Internet Explorer (navigating, waiting for documents to complete, file downloads, clicking, form filling and extracting) use the IEAuto library which comes with WebRecorder, an optional Macro Scheduler add-on.

See also: IETagEvent 140, IETagEventByAttrib 141, IEGetTagsByAttrib 140, IEWait 142

Example

```
//get all input tags from amazon
IEGetTags>amazon.co.uk,INPUT,O,tags

//find the search box (name=field-keywords)
Let>p=0
Let>k=0
While>p=0
  Let>k=k+1
  Let>tag=tags_%k%
  Pos>name=field-keywords,tag,1,p
EndWhile

//Enter a search term in it ...
IETagEvent>amazon.co.uk,INPUT,k,,Grisham

//Find the Go button
Let>p=0
Let>k=0
While>p=0
  Let>k=k+1
  Let>tag=tags_%k%
  Pos>alt=Go,tag,1,p
EndWhile

//click it ...
IETagEvent>amazon.co.uk,INPUT,k,Click,
```

© 2015 MJT Net Ltd

4.10.11 IEGetTagsByAttrib

IEGetTagsByAttrib>URL,Tagname,Attrib=Value,ReturnType,ArrayVar

This function will return an array of tag values for all tags of the given tag name and attribute value in the IE window whose URL matches or contains the specified URL.

URL: The current URL of the IE window to extract from. This can be a substring. If using a substring and you have multiple IE windows open try to be unique to ensure a correct match.

Tagname: The name of the tags to extract. E.g. "A", "INPUT" or "DIV". This can be any tag type.

Attrib=Value: Specify an identifying attribute and value. E.g.: class=image

ReturnType: Use: "H" to return innerHTML (i.e. the HTML between the opening and closing tags); "O" to return outerHTML (includes the HTML in the tag itself) or "T" for innerText (the text that appears between the opening and closing tags).

ArrayVar: The variable name to store the results in. ArrayVar_Count will contain the number of items returned.

This function can be used to extract data from web pages. Use a loop and RegEx/EasyPatterns or similar to further identify specific elements.

Fore more control of Internet Explorer (navigating, waiting for documents to complete, file downloads, clicking, form filling and extracting) use the IEAuto library which comes with WebRecorder, an optional Macro Scheduler add-on.

See also: IETagEvent 140, IEGetTags 139, IETagEventByAttrib 141, IEWait 142

Example

```
//get an array of result DIVs
IEGetTagsByAttrib>amazon.co.uk,DIV,classname=rslt prod,T,pageResults
```

4.10.12 IETagEvent

IETagEvent>URL,Tagname,Index,Event,Value

Interacts with a given tag in the IE window containing the given URL.

URL: The current URL of the IE window to extract from. This can be a substring. If using a substring and you have multiple IE windows open try to be unique to ensure a correct match.

Tagname: The name of the tag. E.g. "A", "INPUT" or "DIV". This can be any tag type.

Index: The numeric index (starting at 1) of the tag. Use IEGetTags to return a list of tags and identify the index you need.

Event: Presently supported: "click" and "focus"

© 2015 MJT Net Ltd

Value: To set the value of an input field specify the new value here.

To determine the Index look at the source of the page or use the IEGetTags function. If the index changes use IEGetTags and a loop with RegEx/EasyPatterns/Position to search each tag to determine which is required based on some identifying attributes.

Fore more control of Internet Explorer (navigating, waiting for documents to complete, file downloads, clicking, form filling and extracting) use the IEAuto library which comes with WebRecorder, an optional Macro Scheduler add-on.

See Also: IEGetTags 139, IEGetTagsByAttrib 140, IETagEventByAttrib 141, IEWait 142

For example see IEGetTags 139

4.10.13 IETagEventByAttrib

IETagEventByAttrib>URL,Tagname,Attrib=Value,Event,Value

Interacts with a given tag in the IE window containing the given URL.

URL: The current URL of the IE window to extract from. This can be a substring. If using a substring and you have multiple IE windows open try to be unique to ensure a correct match.

Tagname: The name of the tag. E.g. "A", "INPUT" or "DIV". This can be any tag type.

Attrib=Value: An identifying attribute and corresponding value. E.g. class=image

Event: Presently supported: "click" and "focus"

Value: To set the value of an input field specify the new value here.

To determine the Index look at the source of the page or use the IEGetTags function. If the index changes use IEGetTags and a loop with RegEx/EasyPatterns/Position to search each tag to determine which is required based on some identifying attributes.

Fore more control of Internet Explorer (navigating, waiting for documents to complete, file downloads, clicking, form filling and extracting) use the IEAuto library which comes with WebRecorder, an optional Macro Scheduler add-on.

See Also: IEGetTags 139, IEGetTagsByAttrib 140, IETagEvent 140, IEWait 142

Example

```
//Search for an author on Amazon:
IETagEventByAttrib>amazon.co.uk,INPUT,Name=fields-keywords,,Grisham

//click Go
IETagEventByAttrib>amazon.co.uk,INPUT,Alt=Go,Click,
```

© 2015 MJT Net Ltd

4.10.14 IEWait

IEWait>url

Waits until the IE browser window matching the specified URL is no longer reporting busy.

See Also: IEGetTags 139, IEGetTagsByAttrib 140, IETagEvent 140, IETagEventByAttrib 141

4.10.15 RetrievePOP3

RetrievePOP3>server,username,password,output_path

Not supported in Macro Scheduler Lite.

Retrieves mail from a POP3 server and stores new mail messages in the folder specified in output_path.

server: the name/address of the POP3 server
username: the POP3 account username
password: the POP3 account password
output_path: a local directory where message files should be stored

Unencoded messages and headers are stored as MSGn.TXT where n is an incremented suffix number. e.g. MSG1.TXT, MSG2.TXT etc. Text body parts will be stored as MSGn_BODYn.TXT and attachments will be stored as MSGn_attachment_filename

The command returns two variables:

POP3_MSGFILES: A semicolon delimited list of files downloaded. Use Separate to explode and retrieve file count.
POP3_RESULT: Contains the last response, such as errors, returned by the server.

Optional variables are:

POP3_STATUS: Set to 0 to suppress the status window. Default is 1.
POP3_PORT: The POP3 port number. Default is 110.
POP3_TIMEOUT: Timeout in seconds. Default is 5.
POP3_DELETE: Set to 1 to delete messages from the server after downloading. Default is 0.
POP3_MSGSIZELIMIT: Size limit in Kb. Messages over this limit will not be downloaded. Default is 0 (no size limit).
POP3_SSL: Set to 1 to use SSL for secure connections. To use SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html and http://www.mjtnet.com/blog/2012/10/03/sendingretrieving-emails-via-gmail/

Abbreviation: RET
See also: SMTPSendMail 143

Example

```
Let>POP3_STATUS=1
Let>POP3_MSGSIZELIMIT=3
Let>POP3_TIMEOUT=5
RetrievePOP3>mail.server.com,username1,password,d:\emailfiles
```

© 2015 MJT Net Ltd

```
MessageModal>POP3_RESULT
Separate>POP3_MSGFILES,;,MsgFiles
MessageModal>Files Downloaded: %MsgFiles_count%%CRLF%File List: %POP3_MSGFILES%
```

4.10.16 SMTPSendMail

SMTPSendMail>recipients,server,from_address,from_name,subject,body,attachments

Not supported in Macro Scheduler Lite.

Sends email via an SMTP server to the recipients entered.

recipients : Include one or more email addresses, separated by semicolons (;).
server : The name or IP address of your SMTP server.
from_address : Your email address, or the email address you want the message to come from.
from_name : Your real name, or a string to appear in the from name field.
subject : The subject line.
body : The body text of the message (default is text, to send HTML set SMTP_HTMLBODY to 1).
attachments : if sending files include each file separated by semicolons (;). If no attachments end the line with a comma.

By default when this command is in use a small status window will appear at the top left of the screen. You can stop this window being displayed by setting SENDMAIL_STATUS to 0.

To send HTML set SMTP_HTMLBODY to 1.

To include CC or BCC email addresses set the SMTP_CCLIST and SMTP_BCCLIST variables. Separate multiple email address with semicolons.

If you are using an SMTP server that requires authentication, you can enable authentication by setting SMTP_AUTH to 1, and then setting SMTP_USERID and SMTP_PASSWORD to the username and password that you need to connect to the SMTP server. e.g:

Let>SMTP_AUTH=1
Let>SMTP_USERID=myuser
Let>SMTP_PASSWORD=frogslegs

To enable return receipt of the email, set SMTP_RECEIPT to 1.

By default the port used by SMTPSendMail is 25. If you need to change the port number set the value of SMTP_PORT.

The result of the sendmail operation is placed into the variable SMTP_RESULT. The format of this response depends on the SMTP server being communicated with. If successful the value will **contain** the result code 250. Otherwise it will contain the appropriate error message.

SMTP_TIMEOUT can be used to set a connection timeout in milliseconds. This timeout relates to the initial connection to the SMTP server.

To use SSL set SMTP_SSL to 1. To use SSL you will first need to install the OpenSSL libraries. See:
http://www.openssl.org/related/binaries.html and http://www.mjtnet.com/blog/2012/10/03/sendingretrieving-emails-via-gmail/

Abbreviation : SMT
See also: RetrievePOP3 142

© 2015 MJT Net Ltd

Example

```
Let>SENDMAIL_STATUS=1
Let>subject=Test Message
Let>me=myname@myplace.com
Let>myname=Mr Smith
Let>recipients=mickey@disney.com;someone@somewhere.com
Input>body,Enter your message:
SMTPSendMail>recipients,post.mail.com,me,myname,subject,body,
Message>Result of SendMail: %SMTP_RESULT%
```

4.10.17 TelnetClose

TelnetClose>session_id

Not supported in Macro Scheduler Lite.Not supported in Macro Scheduler Lite.

Terminates a Telnet session. Provide the session_id returned by a call to TelnetConnect 144.

The TELNET_SESSIONLOG variable can be used at any time to see all session activity.

Abbreviation: TNQ
See also: TelnetConnect 144, TelnetSend 145, TelnetWaitFor 145

Example

```
TelnetConnect>my.domain.com,23,hTN
TelnetWaitFor>hTN,Press any key to continue,5,r
TelnetSend>hTN,a
TelnetWaitFor>hTN,login,5,r
TelnetSend>hTN,administrator%CR%
TelnetWaitFor>hTN,password,5,r
TelnetSend>hTN,monsoon%CR%
TelnetWaitFor>hTN,>,5,r
TelnetSend>hTN,dir%CR%
TelnetWaitFor>hTN,>,5,dirlist
MessageModal>dirlist
TelnetClose>hTN
```

4.10.18 TelnetConnect

TelnetConnect>host,port,session_id

Not supported in Macro Scheduler Lite.

Connects to a Telnet server. Host is the server name or address of the server, port the port number. The standard Telnet port is 23. Returns a Session ID to be used with other Telnet commands to manipulate the session.

The TELNET_SESSIONLOG variable can be used at any time to see all session activity.

Abbreviation: TNC

© 2015 MJT Net Ltd

See also: TelnetClose 144, TelnetSend 145, TelnetWaitFor 145

Example

```
TelnetConnect>my.domain.com,23,hTN
TelnetWaitFor>hTN,Press any key to continue,5,r
TelnetSend>hTN,a
TelnetWaitFor>hTN,login,5,r
TelnetSend>hTN,administrator%CR%
TelnetWaitFor>hTN,password,5,r
TelnetSend>hTN,monsoon%CR%
TelnetWaitFor>hTN,>,5,r
TelnetSend>hTN,dir%CR%
TelnetWaitFor>hTN,>,5,dirlist
MessageModal>dirlist
TelnetClose>hTN
```

4.10.19 TelnetSend

TelnetSend>session_id,text

Not supported in Macro Scheduler Lite.

Sends text to a Telnet session. session_id is a session ID returned by a call to TelnetConnect 144.

To send a carriage return use %CR%.

The TELNET_SESSIONLOG variable can be used at any time to see all session activity.

Abbreviation: TNS
See also: TelnetClose 144, TelnetConnect 144, TelnetWaitFor 145

Example

```
TelnetConnect>my.domain.com,23,hTN
TelnetWaitFor>hTN,Press any key to continue,5,r
TelnetSend>hTN,a
TelnetWaitFor>hTN,login,5,r
TelnetSend>hTN,administrator%CR%
TelnetWaitFor>hTN,password,5,r
TelnetSend>hTN,monsoon%CR%
TelnetWaitFor>hTN,>,5,r
TelnetSend>hTN,dir%CR%
TelnetWaitFor>hTN,>,5,dirlist
MessageModal>dirlist
MessageModal>TELNET_SESSIONLOG
TelnetClose>hTN
```

4.10.20 TelnetWaitFor

TelnetWaitFor>session_id,text,timeout,response

Not supported in Macro Scheduler Lite.

© 2015 MJT Net Ltd

Waits for text in a Telnet session. Specify a session ID returned by a call to TelnetConnect 144.

Enter a timeout in seconds. If the command times out response will be set to TIMEOUT otherwise response will return session output since the last TelnetSend.

The TELNET_SESSIONLOG variable can be used at any time to see all session activity.

Abbreviation: TNW
See also: TelnetClose 144, TelnetConnect 144, TelnetSend 145

Example

```
TelnetConnect>my.domain.com,23,hTN
TelnetWaitFor>hTN,Press any key to continue,5,r
TelnetSend>hTN,a
TelnetWaitFor>hTN,login,5,r
TelnetSend>hTN,administrator%CR%
TelnetWaitFor>hTN,password,5,r
TelnetSend>hTN,monsoon%CR%
TelnetWaitFor>hTN,>,5,r
TelnetSend>hTN,dir%CR%
TelnetWaitFor>hTN,>,5,dirlist
MessageModal>dirlist
MessageModal>TELNET_SESSIONLOG
TelnetClose>hTN
```

4.11 Conditional Commands (If...)

4.11.1 If

If>expression[,true_label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

Evaluates **expression**. If the expression is true the first statements are executed. If the expression is false the statements after Else are executed. Else is optional. IF blocks can be nested. When label names are provided execution of the script jumps to the specified labels or subroutine names. If label names are specified Else and Endif are ignored and are unnecessary.

The **expression** can be *simple (legacy)*, or *complex*.

Complex Expressions

The expression must be contained within curly braces "{" and "}". String literals must be delimited with double quotes ("), e.g.:
"string". Variables must be delimited with % symbols, e.g.: %VarA%.

Several types of operators and functions can be used with complex expressions. For more details see Complex Expressions 54.

Simple Expressions

© 2015 MJT Net Ltd

Simple expressions can contain only two parts separated by one of the following operators:

= Equals
> Greater than
< Less than
<> Not Equal

Values in the expression can be numeric or string values, or variables containing such values. Spaces must not be used as simple expressions do not require string delimiters.

See also: IfNot 201, Label 202, Goto 200, IfFileChanged 124, IfFileExists 124, IfWindowOpen 151, Subroutines 197

Example

```
Let>a=5
//simple expression only
IF>a=5
    //do something
ELSE
    //do something else
ENDIF

//complex expression:
IF>{(%a% = 5) AND (%VarA% = "allen")}
   //do something
ELSE
   //do something else
ENDIF

//using labels
Label>Start
..
..
..
If>a<b,Start
```

4.11.2 IfDirExists

IfDirExists>directory_name[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If directory_name exists the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the directory specified in directory_name exists. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

© 2015 MJT Net Ltd

Abbreviation : IFD
See also: Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfFileExists 124, Subroutines 197

Example

```
IfDirExists>c:\myfolder\temp
  MessageModal>temp folder exists - good
Else
  CreateDir>c:\myfolder\temp
Endif
```

4.11.3 IfFileChanged

IfFileChanged>filename,range[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If the given file's date is in the range of days specified the first statements are executed, otherwise the else statements are executed. Alternatively, instead of Else/Endif specify label names.

When label names are specified execution jumps to the specified label if the given file's date is in the range of days specified. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

Abbreviation : IFC
See also: Label 202, Goto 200, IfWindowOpen 151, IfFileExists 124, If 146, Subroutines 197

Example

To see if test.txt is less than 30 days old :

```
IfFileChanged>test.txt,<30
  MessageModal>test.txt is less than 30 days old
Endif
```

4.11.4 IfFileExists

IfFileExists>filename[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If filename exists the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the file specified in filename exists. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified

© 2015 MJT Net Ltd

Else and Endif are ignored and are unnecessary.

Abbreviation : IFE
See also: Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfDirExists 123, Subroutines 197

Example

```
IfFileExists>c:\myfolder\myfile.txt
  ReadFile>c:\myfolder\myfile.txt,fileContents
Endif
```

4.11.5 IfNot

IfNot>expression[,true_label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

The reverse of If 200. Evaluates **expression**. If the expression is not true the first statements are executed. If the expression is true the statements after Else are executed. Else is optional. IF blocks can be nested. When label names are provided execution of the script jumps to the specified labels or subroutine names. If label names are specified Else and Endif are ignored and are unnecessary.

For more information on how to use this please see If 200.

4.11.6 IfNotDirExists

IfNotDirExists>directory_name[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If directory_name does not exist the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the directory specified in directory_name does not exist. If a second false label is specified, execution will jump to that label if the expression resolves false (directory does exist). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

See also: IfDirExists 123, Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfFileExists 124, Subroutines 197

Example

```
IfNotDirExists>c:\myfolder\temp
  CreateDir>c:\myfolder\temp
Endif
```

© 2015 MJT Net Ltd

4.11.7 IfNotFileChanged

IfNotFileChanged>filename,range[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If the given file's date is not in the range of days specified the first statements are executed, otherwise the else statements are executed. Alternatively, instead of Else/Endif specify label names.

When label names are specified execution jumps to the specified label if the given file's date is not in the range of days specified. If a second false label is specified, execution will jump to that label if the expression resolves false (file date IS in range). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

See also: IfFileChanged 124, Label 202, Goto 200, IfWindowOpen 151, IfFileExists 124, If 146, Subroutines 197

Example

To see if test.txt is not less than 30 days old :

```
IfNotFileChanged>test.txt,<30
  MessageModal>file must be 30 or more days old
Endif
```

4.11.8 IfNotFileExists

IfNotFileExists>filename[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

If filename does not exist the first statements are executed. Otherwise the else statements are executed.

When label names are provided execution jumps to the specified label if the file specified in filename does not exist. If a second false label is specified, execution will jump to that label if the expression resolves false (the file exists). Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return when the subroutine has completed. If label names are specified Else and Endif are ignored and are unnecessary.

See also: IfFileExists 124, Label 202, Goto 200, IfWindowOpen 151, IfFileChanged 124, If 146, IfDirExists 123, Subroutines 197

Example

```
IfNotFileExists>c:\myfolder\myfile.txt
  WriteLn>c:\myfolder\myfile.txt,wres,This is a New File
```

© 2015 MJT Net Ltd

```
Endif
```

4.11.9 IfNotWindowOpen

IfNotWindowOpen>window_title[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

The opposite of IfWindowOpen 247 but with the boolean logic reversed. See IfWindowOpen 247 for usage.

4.11.10 IfWindowOpen

IfWindowOpen>window_title[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

Checks to see if the specified window is open. If so the first statements are executed. Otherwise the else statements are executed. Optionally, instead of Else and Endif a label or subroutine name can be specified.

When label names are specified the command causes the script to continue from the specified label without running any other lines of code in between. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

The window_title may contain the * symbol at the end to indicate a wildcard.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only

Abbreviation : IfW
See also: IfNotWindowOpen 151, Label 202, Goto 200, IfFileExists 124, IfFileChanged 124, If 146, Subroutines 197

Example

© 2015 MJT Net Ltd

```
IfWindowOpen>Notepad - [Untitled],donotepad
..
..
Label>donotepad

or, with a wildcard :

IfWindowOpen>notepad*,donotepad
..
..
Label>donotepad
```

4.12 Image Recognition

4.12.1 CompareBitmaps

CompareBitmaps>bitmap_file_1,bitmap_file_2,result

Not supported in Macro Scheduler Lite.

Compares two bitmap files and returns the percentage match. If the two files are not of the same dimension then result will be -1.

Abbreviation: CPB
See also: GetScreenRes 155, FindImagePos 152, ScreenCapture 156, WaitScreenImage 157

Example

```
GetScreenRes>ScreenW,ScreenH
ScreenCapture>0,0,ScreenW,ScreenH,d:\current_screen.bmp
CompareBitmaps>d:\required_screen.bmp,d:\current_screen.bmp,match
If>match=100
  MessageModal>Screens Match
Endif
```

4.12.2 FindImagePos

FindImagePos>needle_bitmap,haystack,tolerance,return_offset,X_Array,Y_Array,NumFound [,MatchAlgorithm]

Not supported in Macro Scheduler Lite.

Scans a haystack image looking for occurrences of needle_bitmap. needle_bitmap is usually a small image such as an image of a button or other GUI component captured at design time, and haystack would be a screen image.

It is recommended that the Image Recognition Wizard 101 is used to capture needle images and generate FindImagePos code.

needle_bitmap should be a fully qualified path to a bitmap or png file (see MatchAlgorithm type below

© 2015 MJT Net Ltd

for supported image types).

haystack can be set to one of the following values:

1. A bitmap/png file (e.g. %SCRIPT_DIR%\haystack.bmp) (see MatchAlgorithm type below for supported image types).
2. SCREEN - this causes the entire screen to be scanned
3. WINDOW:title where title is a window title (e.g. WINDOW:Untitled - Notepad)
4. WINDOW:Active Window - to work against the active window

MatchAlgorithm was added in v14 and although it is at the end of the parameter list it should be discussed here as it affects the use and outcome of the remaining parameters. Prior to v14 image recognition was exact and performed a precise comparison of pixels.

To use exact matching set MatchAlgorithm to EXACT.

Version 14 introduces the use of a Correlation Coefficient algorithm. To use this set MatchAlgorithm to CCOEFF. This algorithm uses a template matching system attempting to find similarities in images rather than comparing pixels precisely. It is therefore able to find images even if they are not exactly the same and will therefore cope with differences in bit depth, different versions of images across different versions of Windows and different fonts, etc, making image recognition scripts more portable and less likely to fail should the environment change.

The use of the tolerance parameter varies depending on the MatchAlgorithm used.

CCOEFF:Tolerence should be given a value between 0 and 1 where 1 is a precise match and 0 is the least precise. A value of 0.7 is recommended and should allow for subtle variations in the images and works best for portability. The lower the value the more likely false positives will be observed. With CCOEFF matching only one match (if any) is returned - the most likely candidate. CCOEFF will work with bitmap or png files.

EXACT: For exact matching tolerance specifies the color tolerance. If tolerance is zero the pixel colors must match exactly. Tolerance can be set to a value between 0 and 255. If larger than zero the red, green and blue values of the pixels in bitmap_to_scan are checked to see if they are within the tolerance specified (color value + or - tolerance). Smaller values match less and larger values match more. EXACT matching may return more than 1 match if there are more than 1 exact matches in the target area. EXACT can only work with bitmap files.

Each method has benefits and drawbacks. CCOEFF is more intelligent and more tolerant but is slower and will return only one match. EXACT is faster and can return multiple matches but is precise and therefore less portable and will not cope with changes.

To ensure backward compatibility EXACT matching will be used if MatchAlgorithm is omitted. This ensures old scripts will work as they used to.

return_offset is used to determine which coordinates should be returned and can be one of the following:

0 Top left position of located image within bitmap_to_scan. In other words X, Y is the top left corner of needle_bitmap within haystack
1 Center: X,Y will be set to the center of the located image within the main haystack image.
2 Top right
3 Bottom left
4 Bottom right
5 Middle top
6 Middle bottom

© 2015 MJT Net Ltd

7 Middle left
8 Middle right

X_Array is the name of an array to store the X coordinates of each match. If using CCOEFF matching (the default) only the most likely result is returned and so there will be only one result in X_Array_0 if a match is found. If using EXACT matching multiple results may be returned. The first match is stored in X_Array_0, the second in X_Array_1, etc.

Y_Array is the name of an array to store the Y coordinates of each match. If using CCOEFF matching (the default) only the most likely result is returned and so there will be only one result in X_Array_0 if a match is found. If using EXACT matching multiple results may be returned. The first match is stored in Y_Array_0, the second in Y_Array_1, etc.

NumFound returns the number of matches found. If an error is encountered NumFound will be set to -1. E.g. if the needle image is larger than the haystack image, NumFound will be set to -1.

Note that if haystack is SCREEN or a WINDOW the x and y positions returned are absolute screen positions and can be used directly with MouseMove to move the mouse to the found position.

Advanced (If using EXACT matching): By default FindImagePos now scans ALL pixels in each matching rectangle on needle_bitmap. If you wish to scan only a small set of random pixels (and thereby speed up searching) set FIP_SCANPIXELS to that number, e.g. FIP_SCANPIXELS=100. To set back to all set FIP_SCANPIXELS=ALL.

Abbreviation: FIP
See also: GetScreenRes 155, CompareBitmaps 152, ScreenCapture 156, WaitScreenImage 157

Example

```
FindImagePos>d:\today_button.bmp,SCREEN,0.7,1,X,Y,NumFound,CCOEFF
If>NumFound>0
  MouseMove>X_0,Y_0
Endif
```

4.12.3 GetPixelColor

GetPixelColor>X,Y,result

Returns the pixel color of the specified screen coordinate. Enter the coordinates in X and Y, and specify a variable to store the pixel color in.

Abbreviation : GPC
See also: WaitPixelColor 156, GetRectCheckSum 155, WaitRectChanged 157

Example

```
GetPixelColor>652,355,PC
Message>Pixel Color is : %PC%
```

© 2015 MJT Net Ltd

4.12.4 GetRectCheckSum

GetRectCheckSum>TLX,TLY,BRX,BRY,result_value

When passed the top left and bottom right screen coordinates of a rectangle, this function returns the checksum of that rectangle. The checksum is based on the color and position in the rectangle of each individual pixel. The checksum is returned in the variable result_value.

This command will allow for waiting for a specific part of the screen to become equal to a known graphic. Use the command when the graphic to be compared against is present to determine it's checksum. The example below shows a loop which will cause the script to wait until the rectangle contains the required graphic.

TLX - Top Left corner X coordinate
TLY - Top Left corner Y coordinate
BRX - Bottom Right corner X coordinate
BRY - Bottom Right corner Y coordinate

This function was optimized in version 10 to improve performance and now operates very quickly, taking around 0.1 second to scan a 500x500 area of screen. However, on some systems it may consume a lot of CPU cycles in the process. If you need to force the function to yield to the system more you can set GRC_YIELD to 1, but this will have the side effect of slowing the function down, as it gives over CPU time to other processes.

Abbreviation : GRC
See also: WaitRectChanged 157, WaitPixelColor 156, GetPixelColor 154

Example

```
Label>waitforit
GetRectCheckSum>10,10,15,15,cs
If>cs=6.20066481623195E16,Done
Goto>waitforit
Label>Done
Message>There You Go
```

4.12.5 GetScreenRes

GetScreenRes>Width,Height

Not supported in Macro Scheduler Lite.

Returns the dimensions of the primary monitor.

Abbreviation: GSR
See also: CompareBitmaps 152, FindImagePos 152, ScreenCapture 156, WaitScreenImage 157

Example

```
GetScreenRes>X,Y
```

© 2015 MJT Net Ltd

4.12.6 ScreenCapture

ScreenCapture>X1,Y1,X2,Y2,Filename

Not supported in Macro Scheduler Lite.

Captures a portion of the screen to a .BMP (bitmap) or .JPG (jpeg) file and (by default) to the clipboard.

X1 and Y1 represent the upper left corner of the screen area to be copied.
X2 and Y2 represent the lower right corner of the screen area to be copied.
Filename is the name of the file to create and can end with a .BMP, .PNG or .JPG extension.

ScreenCapture returns a variable called SCREENCAP_RESULT which will contain one of the following values:

0: Successful
1: Invalid file type specified. Valid file types are .BMP, .PNG or .JPG
2: Unable to save image to specified file
3: No filename specified
4: Nonsensical coordinates specified (e.g. top >= bottom or left >= right)

To prevent ScreenCapture saving to the clipboard set the SCREENCAP_CLIPBOARD variable to 0.

Abbreviation: SCP
See also: CompareBitmaps 152, FindImagePos 152, WaitScreenImage 157, GetScreenRes 155

Example

```
ScreenCapture>10,10,200,200,d:\mypic.bmp
```

4.12.7 WaitPixelColor

WaitPixelColor>ColorCode,X,Y,Timeout

This command causes Macro Scheduler to wait until the pixel colour at the specified pixel coordinates changes to the colour specified in ColorCode. If it doesn't change to that colour within the number of seconds specified in Timout, the command stops waiting and the variable WPC_RESULT is set to FALSE. WPC_RESULT is TRUE if the command terminated because the colour changed to the specified colour within the specified time. if Timeout is set to 0, the command will wait indefinitely.

To determine the correct colour code to use, click the drop down menu button next to the cursor position monitor on the macro properties form. From the drop down menu select 'Pixel Color' and the pixel colour of the current mouse cursor position will be added to the display along with the X and Y coordinates. Now you can determine the correct colour code of any pixel on the screen.

Abbreviation : WPC
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitCursorChanged 181, GetPixelColor 154, GetRectCheckSum 155, WaitRectChanged 157

Example

```
WaitPixelColor>16777215,652,355,10
Message>WPC_RESULT
```

© 2015 MJT Net Ltd

4.12.8 WaitRectChanged

WaitRectChanged>TLX,TLY,BRX,BRY,Timeout

This command causes Macro Scheduler to wait until the image bound by the specified pixel coordinates changes. If it doesn't change within the number of seconds specified in Timeout, the command stops waiting and the variable WRC_RESULT is set to FALSE. WRC_RESULT is TRUE if the command terminated because the image changed within the specified time. if Timeout is set to 0, the command will wait indefinitely.

TLX - Top Left corner X coordinate
TLY - Top Left corner Y coordinate
BRX - Bottom Right corner X coordinate
BRY - Bottom Right corner Y coordinate

This function was optimized in version 10 to improve performance and now operates very quickly, taking around 0.1 second to scan a 500x500 area of screen. However, on some systems it may consume a lot of CPU cycles in the process. If you need to force the function to yield to the system more you can set GRC_YIELD to 1, but this will have the side effect of slowing the function down, as it gives over CPU time to other processes.

Abbreviation : WRC
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitCursorChanged 181, WaitPixelColor 156, GetRectCheckSum 155

Example

```
WaitRectChanged>10,10,100,100,10
Message>WRC_RESULT
```

4.12.9 WaitScreenImage

WaitScreenImage>BitmapFile[,Tolerance[,MatchAlgorithm]]

Not supported in Macro Scheduler Lite.

Waits until the image in BitmapFile is found on the screen.

It is recommended to use the Image Recognition Wizard 10 to create WaitScreenImage code.

Returns the number of occurrences found on the screen in the WSI_RESULT variable.

For an explanation of Tolerance and MatchAlgorithm please see FindImagePos 152.
WaitScreenImage is effectively a wrapper for a loop containing FindImagePos.

The system variable WSI_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WSI_TIMEOUT is used, WSI_TIMEDOUT will indicate whether or not the command ended successfully. If it timed out WSI_TIMEDOUT will be set to TRUE otherwise, if the image was found within the time out period it will be set to FALSE. If the command times out WSI_BITMAP will contain the path of BitmapFile. This is useful if using a global event handler for WSI_TIMEDOUT and you nee to know which WaitScreenImage timed out.

Abbreviation: WSI

© 2015 MJT Net Ltd

See also: CompareBitmaps 152, FindImagePos 152, ScreenCapture 156, GetScreenRes 155

Example

```
WaitScreenImage>d:\today_button.bmp
MessageModal>FoundIt
```

4.13 Keyboard Commands

4.13.1 Ascii

Ascii>ASCII_Code[,ASCII_Code[,ASCII_Code[...]]]

Inserts the characters specified by the given ASCII codes into the current application. If sending more than one ASCII code at once, seperate them with commas.

This function is useful for sending non-printable, low, or high ASCII value characters to an editor, which you might otherwise be unable to send using the Send 162 function.

Abbreviation : ASC
See also: Send Character/Text 162

Example

The following example shows three methods of sending the same text value to Notepad.

```
SetFocus>Notepad*
Ascii>77
Ascii>65
Ascii>82
Ascii>67
Ascii>85
Ascii>83
Press Enter
Ascii>77,65,82,67,85,83
Press Enter
Send>MARCUS
```

4.13.2 CapsOff

Switches caps lock off. If Caps lock is already off, no action is taken. If Caps lock is on, it is switched off.

Abbreviation : COF
See also: CapsOn 158

4.13.3 CapsOn

Switches caps lock on. If Caps lock is already on, no action is taken. If Caps lock is off, it is switched on.

Abbreviation : CAP

© 2015 MJT Net Ltd

See also: CapsOff 158

4.13.4 NumOff

Switches Num lock off. If Num lock is already off, no action is taken. If Num lock is on, it is switched off.

Abbreviation : NOF
See also: NumOn 159

4.13.5 NumOn

Switches Num lock on. If Num lock is already on, no action is taken. If Num lock is off, it is switched on.

Abbreviation : NON
See also: NumOff 159

4.13.6 ObjectSendKeys

ObjectSendKeys>handle,keystroke_list

Sends keystrokes directly to the given control. keystroke_list is a comma delimited list of characters or virtual key codes. Virtual key codes should be decimal values preceded by "VK".

For a list of virtual key codes see: http://www.mjtnet.com/vkcodes.htm

Use the Send Keys to Object Wizard to help create code that uses ObjectSendKeys.

Unlike the SendText 162 command ObjectSendKeys sends directly to the control and does not require that the object has focus.

See also: ObjectSendText 159, FindObject 255, SendText 162

Example:

```
//Created by Send Keys to Object Wizard - sends Ctrl+Home, Shift+Ctrl+End, Delete to Notepad's
GetWindowHandle>Untitled - Notepad,hWndParent
FindObject>hWndParent,Edit,,1,hWnd,X1,Y1,X2,Y2,result
ObjectSendKeys>hWnd,CTRL_DN,VK36,CTRL_UP,SHIFT_DN,CTRL_DN,VK35,SHIFT_UP,CTRL_UP,VK46
```

4.13.7 ObjectSendText

ObjectSendText>handle,text

Sends text directly to the given object via keyboard messages.

Unlike the SendText 162 command ObjectSendText sends directly to the control and does not require that the object has focus.

See also: ObjectSendText 159, FindObject 255, SendText 162

© 2015 MJT Net Ltd

Example:

```
//Send text directly to Notepad
GetWindowHandle>Untitled - Notepad,hWndParent
FindObject>hWndParent,Edit,,1,hWnd,X1,Y1,X2,Y2,result
ObjectSendText>hWnd,{"Hello World"}
```

4.13.8 Press ...

All commands starting with Press facilitate the sending of non-character keys. To issue a Press command more than once add the '*' symbol followed by the number of times to issue the command at the end of the line:

Press Key * n

E.g.:

Press Tab * 5

Use SK_DELAY to insert a delay between key presses. Set SK_DELAY to a number of milliseconds to wait between key presses. Th default value for SK_DELAY is 0.

By setting PRESS_ALLOWVARS to 1, Press can accept a variable name in place of the key name.

The following is a complete list of all commands, with explanations where necessary :

Press Backspace
Press Tab
Press Enter
Press Esc

Press F1	Press F13
Press F2	Press F14
Press F3	Press F15
Press F4	Press F16
Press F5	Press F17
Press F6	Press F18
Press F7	Press F19
Press F8	Press F20
Press F9	Press F21
Press F10	Press F22
Press F11	Press F23
Press F12	Press F24

Press Home
Press End
Press Up
Press Down
Press Left
Press Right
Press Page Up
Press Page Down
Press Ins
Press Del
Press Shift
Release Shift

© 2015 MJT Net Ltd

Press LShift	} Left/Right Shift Key
Release LShift	} ONLY in Windows NT and later
Press RShift	} you can distinguish specifically
Release RShift	} between left and right keys.
Press CTRL	
Release CTRL	
Press LCTRL	} Left/Right CTRL Key
Release LCTRL	} ONLY in Windows NT and later
Press RCTRL	} you can distinguish specifically
Release RCTRL	} between left and right keys.
Press ALT	
Release ALT	
Press LALT	} Left/Right ALT Key
Release LALT	} ONLY in Windows NT and later
Press RALT	} you can distinguish specifically
Release RALT	} between left and right keys.
Press ALTGR	
Release ALTGR	
Press CAPS	has the effect of toggling caps lock.
Press Num Lock	
Press Scroll Lock	
Press NP0	0 on Number Pad
Press NP1	etc
Press NP2	
Press NP3	
Press NP4	
Press NP5	
Press NP6	
Press NP7	
Press NP8	
Press NP9	
Press NP Add	Num pad operator keys
Press NP Subtract	etc
Press NP Multiply	
Press NP Divide	
Press NP Decimal	
Press NP Enter	
Press LWinKey	}
Press RWinKey	} Windows Keys
Press MenuKey	}
Press Print Screen	
Press Space	Same as Send>space or Send>{" "} or Send>

There have been reports that Press CTRL, Press SHIFT and Press ALT sometimes fail when running Macro Scheduler under a Citrix environment. This can be fixed by setting SK_LEGACY to 1 prior to issuing these keystroke commands. Some DOS applications also need SK_LEGACY set to 1.

4.13.9 Release ...

Some keys that can be pressed must also be released. This facilitates holding down a key while another is pressed, such as with the ALT key for instance.

For example, to exit a program you would press ALT and F together to activate the File menu, followed by the X key to select the Exit option. To simulate this in a script you would Press ALT, then send the text FX and finally Release ALT. This would appear in the script window as :

© 2015 MJT Net Ltd

Press ALT
Send Character/Text>FX
Release ALT

By setting PRESS_ALLOWVARS to 1, Release can accept a variable name in place of the key name.

The following release key commands exist :

Release ALT
Release LALT
Release RALT
Release ALTGR

Release CTRL
Release LCTRL
Release RCTRL

Release Shift
Release LShift
Release RShift

Release LWinKey
Release RWinKey

4.13.10 ScrollOff

Switches Scroll lock off. If Scroll lock is already off, no action is taken. If Scroll lock is on, it is switched off.

Abbreviation : SOF
See also: ScrollOn 162

4.13.11 ScrollOn

Switches Scroll lock on. If Scroll lock is already on, no action is taken. If Scroll lock is off, it is switched on.

Abbreviation : SON
See also: ScrollOff 162

4.13.12 SendText

SendText>text_to_send

This command sends the specified text to the window that currently has the focus. See SetFocus 249.

It is possible to slow down the speed at which each character in the string is sent by using the SK_DELAY variable. Set this to the number of milliseconds to pause between each individual character. e.g.: Let>SK_DELAY=10. The default for SK_DELAY is 0.

© 2015 MJT Net Ltd

Note that this command sends keystrokes and therefore any keys sent will be subject to the current state of modifier keys and the caps lock key. To ignore the caps lock state and send the text as entered set SK_IGNORECAPS to 1. (Let>SK_IGNORECAPS=1).

This command is commonly abbreviated to Send.

Abbreviation : Sen (Send)
See also: Ascii 158

Examples

```
SendText>Here Is Some Text ...
```

The following example simulates pressing ALT-FX, a standard key combination for closing a program :

```
Press ALT
SendText>fx
Release ALT
```

Variables can be used :

```
Let>SomeText=Hello World
Send Character/Text>SomeText
```

4.13.13 WaitKeyDown

WaitKeyDown>key_to_wait_for

WaitKeyDown pauses execution until the specified key is pressed. For an ordinary character key specify the character of the key. For other keys use the virtual key code preceeded by VK:

WaitKeyDown>H - waits for the H key to be pressed

WaitKeyDown>VK101 - waits for virtual key code 101 (Numpad 5) to be pressed.

A list of virtual key codes can be found here: http://www.mjtnet.com/vkcodes.htm

Abbreviation : WKD
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitCursorChanged 181, WaitPixelColor 156, WaitReady 209

4.14 Loops and Branching

4.14.1 EndWhile

EndWhile

Use in conjunction with While to create a loop. The code between While and EndWhile will repeat while condition is true.

Note that the condition is evaluated at the start of the loop and therefore at the start of each iteration.

© 2015 MJT Net Ltd

See also: While 167, Repeat 208

Example

```
Let>x=0
While>x<10
  Let>x=x+1
  MessageModal>x
EndWhile
```

4.14.2 Gosub

Gosub>Subroutine_Name[,value1[,value2[,...]]]

Branches to the specified subroutine (SRT). When that the subroutine returns, processing continues at the next line after the Gosub command.

Subroutines can be nested. Subroutines can be located anywhere within the script.

If values are passed on the Gosub line variables will be created to contain these values. The variables are named with the following format: SubroutineName_Var_1, Subroutine_Var_2, ... Subroutine_Var_n

See also SRT 197, END 198

Example

```
Gosub>EnterName
Gosub>CloseApp

//// Subroutines Below ////

SRT>FocusApp
   SetFocus>Data Entry Screen
End>FocusApp

SRT>EnterName
   Gosub>FocusApp
   Send>Firstname
   Press Tab
   Send>Surname
   Press Enter
End>EnterName

SRT>CloseApp
   Gosub>FocusApp
   Press ALT
   Press F4
   Release ALT
End>CloseApp
```

© 2015 MJT Net Ltd

4.14.3 Goto

Goto>Label_Name

Causes execution to continue at the specified label, missing any commands in between. If the label does not exist an error message will be displayed.

Label_Name can be or include a variable name.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also: Label 202

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

4.14.4 Label

Label>Label_Name

Marks a point in the script to allow execution to be passed to that point by the Goto, and If.. commands.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also: Goto 200, If 146, IfWindowOpen 151, IfFileExists 124, IfFileChanged 124

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

© 2015 MJT Net Ltd

4.14.5 Repeat

Repeat>variable

Use in conjunction with Until. Iterates the code from the Repeat statement to the Until statement until the specified expression in the Until statement becomes true. Until can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Alternatively Until can take a full complex expression 54 (specified between curly braces).

Until will loop back to the last Repeat that has the same counter variable.

See also: Until 209

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
```

4.14.6 SkipLabel

SkipLabel>LabelName

SkipLabel is used within a subroutine to tell the script to jump to a label *after* the subroutine has ended. It is bad practice to jump out of a subroutine using a Goto statement - as the subroutine may then never end, causing problems with, for example, OnEvent handlers. In some cases however it is useful to cause the script to jump to a label once the subroutine has finished. This is what SkipLabel does.

Note that SkipLabel is NOT supported by OnEvent subroutines.

Example

```
Let>x=0
Repeat>x
  Let>x=x+1
  GoSub>ExampleSubroutine
Until>x=100

SRT>ExampleSubroutine
  If>x=5
    SkipLabel>EndScript
  Endif
```

© 2015 MJT Net Ltd

```
  //this bit still gets executed
  Let>y=x
END>ExampleSubroutine

Label>EndScript
MessageModal>y
```

4.14.7 Until

Until>variable,finalvalue

Use in conjunction with Repeat. Iterates the code from the Repeat statement to the Until statement until the specified expression in the Until statement becomes true. Until can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Until will loop back to the last Repeat that has the same counter variable.

See also: Repeat 208

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
```

4.14.8 While

While>condition

Use in conjunction with EndWhile to create a loop. The code between While and EndWhile will repeat while condition is true.

Note that the condition is evaluated at the start of the loop and therefore at the start of each iteration.

While can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Alternatively While can take a full complex expression 54 (specified between curly braces).

© 2015 MJT Net Ltd

See also: EndWhile 163, Repeat 208

Example

```
Let>x=0
While>x<10
  Let>x=x+1
  MessageModal>x
EndWhile
```

4.15 Messages

4.15.1 Ask

Ask>prompt,result_variable

Displays a Yes, No dialog box with the specified prompt. If the user presses 'Yes', result_variable is set to 'YES', else result_variable becomes 'NO'.

The result is returned in upper case.

prompt can be a variable containing the prompt to display.

It is possible to set a timeout by setting the value of ASK_TIMEOUT before issuing the Ask command. ASK_TIMEOUT takes a number of milliseconds. If the Ask box is still active after the timeout has elapsed it will be closed and the result set to YES.

See also: Input 168, Message 169, MessageModal 170

Example

```
Ask>Do you want to continue ?,continue
```

4.15.2 Input

Input>variable,prompt[,default_value]

Displays a dialog box to request information from the user. The dialog box displays the prompt specified in prompt and accepts input into variable. Optionally, a default value can be specified.

The Input box now also has a file browse button, making it useful for accepting filenames from the user. The file browse button can be hidden by setting INPUT_BROWSE to 0 (Let>INPUT_BROWSE=0). Set INPUT_BROWSE to 2 to display a folder browse dialog for locating folders instead of files.

If the Input dialog is cancelled (cancel is pressed) Input returns an empty string.

prompt can be a variable, containing the prompt to display.

It is possible to mask the value entered by the user with asterisks by setting the

© 2015 MJT Net Ltd

INPUT_PASSWORD variable to 1. Set to zero for default behaviour.

It is possible to set a timeout by setting the value of INPUT_TIMEOUT before issuing the Input command. INPUT_TIMEOUT takes a number of milliseconds. If the Input box is still active after the timeout has elapsed it will be closed and the result set to an empty string.

Abbreviation : Inp
See also: Ask 168, Message 169, MessageModal 170

Example

```
Input>name,Please enter your name ...
```

4.15.3 Message

Message>message_text

Displays a message box containing the text specified in message_text. In order that execution of the script can continue, these message boxes are not modal. This means they can be used to display information even if the script is not being run interactively. For modal message boxes, use MessageModal 170.

To make a message box stay on top, set the variable MSG_STAYONTOP to 1. Likewise, setting MSG_CENTERED to 1 will ensure that the message box is always shown centrally on the screen.

You can set/get the height and width of the message box using the variables MSG_HEIGHT and MSG_WIDTH. Set the position of the message box using the variables MSG_XPOS and MSG_YPOS.

Abbreviation : MSG
See also: MessageModal 170, Input 168, Ask 168

Example

```
Message>Hello World!
```

or with variables ..

```
Let>mymsg=Hello World!
Message>mymsg
```

To force a new line in a message box, use the CRLF system variable :

```
Message>Hello World %CRLF% %CRLF%End Message.
```

This would display :

Hello World

End Message.

To make the message box stay on top (above all other windows) :

```
Let>MSG_STAYONTOP=1
Message>Hello World
```

© 2015 MJT Net Ltd

4.15.4 MessageModal

MessageModal>message_text

Displays a message box containing the text specified in message_text. With this command the Message box is modal. This means that the script will not continue until OK is pressed. For non modal message boxes use Message 169.

To make a message box stay on top, set the variable MSG_STAYONTOP to 1. Likewise, setting MSG_CENTERED to 1 will ensure that the message box is always shown centrally on the screen.

You can set/get the height and width of the message box using the variables MSG_HEIGHT and MSG_WIDTH. Set the position of the message box using the variables MSG_XPOS and MSG_YPOS.

Abbreviation : MDL
See also: Message 169, Input 168, Ask 168

Example

```
MessageModal>Hello World!
```

or with variables ..

```
Let>mymsg=Hello World!
MessageModal>mymsg
```

To embed a variable within a string:

```
Let>forename=Fred
MessageModal>Hello %forename%
```

4.16 Miscellaneous

4.16.1 AddTrayHandler

AddTrayHandler>Name,Event,Subroutine

Assigns a subroutine to a system tray event. Events can be one of:

OnClick
OnDblClick
OnMouseDown
OnMouseMove
OnMouseUp

See also: AddTrayIcon 171, DelTrayIcon 173
Abbreviation: ATH

Example

```
AddTrayIcon>%SCRIPT_DIR%\tray.ico,MyTrayIcon,Click Me To Open
AddTrayHandler>MyTrayIcon,OnClick,DoTrayClick
```

© 2015 MJT Net Ltd

```
...

SRT>DoTrayClick
  Show>Dialog1,res
End>DoTrayClick
```

4.16.2 AddTrayIcon

AddTrayIcon>IconFileName,Name,Hint

Creates an icon in the system tray using the .ico file specified. Name should be a unique name which can be used to reference the tray icon in the DelTrayIcon and AddTrayHandler functions. Set Hint to the hint text to display when the mouse is moved over the tray.

See also: DelTrayIcon 173, AddTrayHandler 170, ModTrayIcon 178
Abbreviation: ATI

Example

```
AddTrayIcon>%SCRIPT_DIR%\tray.ico,MyTrayIcon,Click Me To Open
AddTrayHandler>MyTrayIcon,OnClick,DoTrayClick

...

SRT>DoTrayClick
  Show>Dialog1,res
End>DoTrayClick
```

4.16.3 ArrayCount

ArrayCount>array_name,result[,suffixformat]

Returns in result the number of items in a simple one dimensional array.

Suffixformat is used to specify the format used for the array suffix as array variables in Macro Scheduler can take any format so desired. If omitted suffixformat is set to "_n" where n is the numeric placeholder. This would match an array with the format array_1, array_2, etc. An array format that looks like array[1], array[2] ... array[n] has the suffix format "[n]" (without the quotes)

ArrayCount works with numeric array indexes only. Only arrays starting at index 0 or 1 will be recognised. ArrayCount assumes that the array index advances by 1 for each element. If there are any gaps ArrayCount will stop at the last element before the gap.

See also: ArraySort 172, Arrays 265
Abbreviation: ARC

Example:

```
GetFileList>%SCRIPT_DIR%\*.*,MyFiles
Separate>MyFiles,;,FilesArray
ArrayCount>FilesArray,numItems
```

© 2015 MJT Net Ltd

4.16.4 ArrayDim

ArrayDim>array_name,num_elements

Creates an empty array with specified number of elements. If the array already exists and num_elements is larger than the existing number of elements the array is lengthened with extra empty elements.

See also: Arrays 265
Abbreviation: ARR

Example:

```
ArrayDim>Names,3
Let>Names_1=Fred
Let>Names_2=Sally
Let>Names_3=John
```

4.16.5 ArraySort

ArraySort>array_name,[,suffixformat]

Sorts the given array in ascending alphanumeric order.

Suffixformat is used to specify the format used for the array suffix as array variables in Macro Scheduler can take any format so desired. If omitted suffixformat is set to "_n" where n is the numeric placeholder. This would match an array with the format array_1, array_2, etc. An array format that looks like array[1], array[2] ... array[n] has the suffix format "[n]" (without the quotes).

ArraySort works with numeric array indexes only. Only arrays starting at index 0 or 1 will be recognised. ArraySort assumes that the array index advances by 1 for each element. If there are any gaps ArraySort will stop at the last element before the gap.

See also: ArrayCount 171, Arrays 265
Abbreviation: ARS

Example:

```
GetFileList>%SCRIPT_DIR%\*.*,MyFiles
Separate>MyFiles,;,FilesArray
ArraySort>FilesArray
```

4.16.6 ColorToRGB

ColorToRGB>color_code,red,green,blue

Returns the red, green and blue values of a given color code. Does the opposite of the RGB 181 function.

See also: RGB 181

© 2015 MJT Net Ltd

Example:

```
ColorToRGB>6554367,r,g,b
MessageModal>%r%,%g%,%b%
```

4.16.7 DelArray

DelArray>array_name,[,suffixformat]

Deletes the specified array. Deleting an array removes all array element variables from the variable table such that they are no longer available for use.

See also: DelVariable 173

Example:

```
DelArray>Names
```

4.16.8 DelTrayIcon

DelTrayIcon>Name

Deletes the specified icon from the system tray. The icon must have been created with the AddTrayIcon function.

See also: AddTrayIcon 171, AddTrayHandler 170, ModTrayIcon 178
Abbreviation: DTI

Example

```
AddTrayIcon>%SCRIPT_DIR%\tray.ico,MyTrayIcon,Click Me To Open

...

Label>end_script
DelTrayIcon>MyTrayIcon
```

4.16.9 DelVariable

DelVariable>Variable_Name

Deletes the specified variable from the variable table. Once deleted a variable will no longer be available and any reference to it will yield the literal string instead.

See also: DelArray 173

Example:

```
DelVariable>x
```

© 2015 MJT Net Ltd

4.16.10 BlockInput

BlockInput>block

Blockinput prevents or enables keyboard and mouse input from the user.

Set block to 1 to block input from the user.
Set block to 0 to enable input.

Be aware that this prevents all user input apart from CTRL-ALT-DEL. If you need to stop a running script that has blocked input press CTRL-ALT-DEL and then stop the script.

Under Windows Vista or above this function will only work if Macro Scheduler is running elevated as Admin (right click the Macro Scheduler icon and select Run as Admin).

Abbreviation: BLO

Example:

```
//Block input from user
BlockInput>1

Run>Notepad
WaitWindowOpen>Notepad*
Send>Hello World

BlockInput>0
```

4.16.11 ExportData

ExportData>Data_Label,Filename

Exports binary data from the script to the specified file. Use this in conjunction with Tools/Import Binary File in the editor. This allows you to include binary files in your scripts and export them for use at run time.

Abbreviation: EXP

Example

```
ExportData>CALC.DLL_DATA,%SCRIPT_DIR%\calc.dll

CALC.DLL_DATA:
4D5A50000200000004000F00FFFF0000B80000000000000...
```

4.16.12 GetEnvVar

GetEnvVar>EnvironmentVariable,LocalVariable

Retrieves the value of the specified environment variable into the local variable given in LocalVariable.

Abbreviation: GEV

© 2015 MJT Net Ltd

See also: SetEnvVar 181

Example:

```
GetEnvVar>TMP,myTMPDir
MessageModal>myTMPDir
```

4.16.13 KillProcess

KillProcess>process

Terminates the given process. Process can be a module name (e.g. notepad.exe) or a numeric process ID.

See also: WaitProcessExists 182, WaitProcessTerminated 182, ProcessExists 179
Abbreviation: Kil

Example

```
ProcessExists>notepad.exe,bNotepadExists
If>bNotepadExists=True
  KillProcess>notepad.exe
Endif
```

4.16.14 LibFree

LibFree>module_handle

Not supported in Macro Scheduler Lite.

Frees a library previously loaded with LibLoad.

Abbreviation : LFR
See also: LibLoad 177, LibFunc 175

Examples

```
LibLoad>user32,hDll
Let>WIN_USEHANDLE=1
GetActiveWindow>win,x,y
LibFunc>hDll,GetWindowTextLengthA,wtlen,win
Let>buffer_SIZE=wtlen
LibFunc>hDll,GetWindowTextA,gwt,win,buffer,wtlen
MessageModal>Window Text: %gwt_2%
LibFree>hDll
```

4.16.15 LibFunc

LibFunc>module,function,resultvar[,[[ref:]var1][,[ref:]var2][,...][,[ref:]varn]]

Not supported in Macro Scheduler Lite.

© 2015 MJT Net Ltd

LibFunc can be used to call DLL functions and is therefore an ideal way of extending the language using custom DLLs or for calling Win32 API functions. (NB: Alternatively, for language extensions, create an ActiveX object and access it using VBScript).

LibFunc takes the following parameters:

module: the DLL module filename to load or a module handle created with a previous call to LibLoad 177
function: the name of the function in the DLL.
resultvar: numeric value returned by the function.
parameter list: See below.

You can specify as many parameters as needed by the DLL function. To pass numeric parameters by reference precede the value with "ref:" (without the quotes). See examples below.

As string variables are actually pointers to memory locations strings are effectively always passed by reference and can be modified by the DLL. String buffer sizes are set to 255 characters by default. To set the size of a string variable to some other value create a parmname_SIZE variable, where parmname is the name of the string variable being passed, set to the length required. See window title example below.

LibFunc determines whether a variable is a numeric integer or a string automatically by looking at it's contents. However, if you need to send a number as a string you can force it to be a string by preceeding the parameter with "str:" (without the quotes).

Note that LibFunc supports long integer and string types only. LibFunc supports ANSI strings only. For Unicode strings use LibFuncW.

As well as returning the function result in resultvar LibFunc creates new variables corresponding to each of the passed parameters with the format resultvar_n where n is the index of the parameter, starting from 1. If the function changed the value of any of the parameters these new variables will contain the new value. Otherwise they will contain the value passed. E.g. if resultvar is set to "answer" and you specified three parameters LibFunc will return four variables called answer, answer_1, answer_2 and answer_3. See examples.

Beware: When you pass data to DLLs you are passing references to memory areas. If you send the wrong type of information or the wrong number of arguments you are likely to cause an access violation error in the DLL which may cause Macro Scheduler to crash. This function is recommended for experienced developers only.

Abbreviation : Lib
See also: LibLoad 177, LibFree 175

Examples

Get Windows System Directory:

```
LibFunc>Kernel32,GetSystemDirectoryA,dir,buffer,255
MessageModal>System Directory: %dir_1%
```

Use ShellExec to open a file/run an application:

```
LibFunc>shell32,ShellExecuteA,r,0,open,notepad,,,1
```

Get free disk space on drive C:

```
LibFunc>kernel32,GetDiskFreeSpaceExA,spacefree,c:\,REF:0,REF:0,REF:0
```

© 2015 MJT Net Ltd

```
Let>used=spacefree_3/1024
Let>free=spacefree_4/1024
MessageModal> Disk space used: %used%Kb %CRLF% Free space: %free%Kb
```

Display a system yes/no message box:

```
Let>YesNo=4
LibFunc>user32,MessageBoxA,r,0,Hello World,Message!,YesNo
If>r=6
  MessageModal>You Pressed Yes
else
  MessageModal>You Pressed No
endif
```

Use the system API Beep function to play sounds:

```
Let>k=200
Repeat>k
  LibFunc>kernel32,Beep,r,k,50
  Let>k=k+10
Until>k,1000
```

Run a function in a custom DLL:

```
LibFunc>C:\mydll\testdll.dll,ChangeText,r,hello
MessageModal>%r% %r_1%
```

Get Active Window Title:

```
Let>WIN_USEHANDLE=1
GetActiveWindow>win,x,y
LibFunc>user32,GetWindowTextLengthA,wtlen,win
Let>buffer_SIZE=wtlen
Let>wtlen=wtlen+1
LibFunc>user32,GetWindowTextA,gwt,win,buffer,wtlen
MessageModal>Window Text: %gwt_2%
```

4.16.16 LibFuncW

LibFuncW>module,function,resultvar[,[[ref:]var1][,[ref:]var2][,...][,[ref:]varn]]

Not supported in Macro Scheduler Lite.

LibFuncW is the Unicode version of LibFunc 175.

4.16.17 LibLoad

LibLoad>module_name,module_handle

Not supported in Macro Scheduler Lite.

Loads the DLL libary specified in module_name and returns a handle to the library which can be used in subsequent LibFunc calls. Libraries loaded with LibLoad should always be freed with a call to LibFree after being used.

© 2015 MJT Net Ltd

Abbreviation : LLD
See also: LibFunc 175, LibFree 175

Examples

```
LibLoad>user32,hDll
Let>WIN_USEHANDLE=1
GetActiveWindow>win,x,y
LibFunc>hDll,GetWindowTextLengthA,wtlen,win
Let>buffer_SIZE=wtlen
Let>wtlen=wtlen+1
LibFunc>hDll,GetWindowTextA,gwt,win,buffer,wtlen
MessageModal>Window Text: %gwt_2%
LibFree>hDll
```

4.16.18 ModTrayIcon

ModTrayIcon>Name,Hint,IconFileName,Visible

Modifies a tray icon created with the AddTrayIcon command. Specify the name of the icon to modify.

Hint can contain either an empty string or a new hint text. If empty the hint is not changed.
IconFileName can contain either an empty tring or a .ico file name. If empty the icon image is not changed
Visible can be True or False to show or hide the icon.

See also: AddTrayIcon 171, AddTrayHandler 170, DelTrayIcon 173

Example

```
ModTrayIcon>MyTrayIcon,Don't Click Me!,,True
```

4.16.19 PlayWav

PlayWav>wav_file

Plays a .WAV sound file.

Abbreviation : Pla

Example

```
PlayWav>c:\windows\media\chimes.wav
```

4.16.20 PopupMenu

PopupMenu>X,Y,ItemList,result

Creates and displays a simple pop up menu at the specified X,Y position and returns the index of the

© 2015 MJT Net Ltd

item selected by the user.

X and Y take a screen X,Y position
ItemList is a semi-colon delimited list of captions
result is the return value and will contain the zero based index of the item the user selected

Example

```
Let>Items=Open;Close;About;Exit
GetCursorPos>X,Y
PopupMenu>X,Y,Items,res
MessageModal>You selected item: %res%
```

4.16.21 ProcessExists

ProcessExists>processname,result

Sets result to True if processname exists, False if not.

See also: WaitProcessExists 182, WaitProcessTerminated 182, KillProcess 175
Abbreviation: Pro

Example

```
ProcessExists>notepad.exe,bNotepadExists
If>bNotepadExists=False
  Run>notepad.exe
Endif
```

4.16.22 PyExec

PyExec>Code,OutputBuffer[,var1,var1,....varn]

Executes the specified python code.

OutputBuffer returns any content written to standard IO. Optionally specify one or more Python variables to be extracted using.

Requires the Python 2.7 DLL to be installed and in the path.

We recommend installing the python27.dll in the same folder as Macro Scheduler or of your compiled macro. Any libraries and runtimes can also be included in lib,libs,dll etc folders at the same level, consistent with the usual Python setup.

We've provided a zip file containing the Python 2.7 runtime environment which you can download from here. Export the contents to the Macro Scheduler program folder, or the folder of a compiled macro (.exe).

Important: Only one Python engine can be active at any one time. This means that scripts running simultaneously cannot use PyExec at the same time. Therefore if PyExec is already running in another script your script will wait until the other script has finished with it before continuing. This does not apply to compiled macros running as .exes as these are running as completely separate processes.

© 2015 MJT Net Ltd

Example

```
Let>url=http://ip.jsontest.com/

/*
python_code:

import urllib2
import json

# grab data from http://ip.jsontest.com/ - see www.jsontest.com
response = urllib2.urlopen('%url%')

# load the json
dict = json.loads(response.read())

# get the ip member
myip = dict["ip"]

# make a nice string representation of the dict
sdict = json.dumps(dict)

# Anything we print to IO is returned in the PYExec output var
print "All Done"
*/

//Load the Python code to a variable
LabelToVar>python_code,pcode

//Run the code and request the values of the sdict and myip variables ...
PYExec>pcode,output,sdict,myip

//Display the IP address
MessageModal>Your pubic IP is: %myip%
```

4.16.23 Random

Random>Range,Result

Returns a random number within the specified range where 0 <= Result < Range.

The seed is set automatically and is stored in the RND_SEED variable. It is possible to set the seed programmatically by modifying the value of RND_SEED.

Result is a variable in which the result is stored.

Abbreviation : RAN

Example

```
Random>6,DiceResult
Let>DiceResult=DiceResult+1
Message>You threw a %DiceResult%
```

© 2015 MJT Net Ltd

4.16.24 RGB

RGB>red,green,blue,result

Creates a color code based on the specified mix of red, green and blue intensities.

Red, Green and Blue can be numbers from 1 to 255. The higher the number the greater the intensity of that color.

Example:

```
RGB>20,50,200,aColor
```

4.16.25 SetEnvVar

SetEnvVar>EnvironmentVariable,Value

Sets the value of the specified environment variable with Value.

Abbreviation: SEV
See also: GetEnvVar 174

Example:

```
SetEnvVar>name,fred
```

4.16.26 WaitCursorChanged

WaitCursorChanged>Timeout

This command causes Macro Scheduler to wait until the cursor of the foreground window changes. If it doesn't change within the number of seconds specified in Timout, the command stops waiting and the variable WCC_RESULT is set to FALSE. WCC_RESULT is TRUE if the command terminated because the foreground window cursor changed within the specified time. if Timeout is set to 0, the command will wait indefinitely.

This command is useful for waiting for applications to become idle. For example, it can be used after initiating some operation in an application that invokes the hourglass cursor, so that you can wait for the application to become idle again.

Abbreviation : WCC
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitPixelColor 156

Example

```
Change Directory>c:\program files\agent\data
Run Program>"c:\program files\agent\agent.exe"
WaitCursorChanged>500
SetFocus>Agent*
```

© 2015 MJT Net Ltd

4.16.27 WaitProcessExists

WaitProcessExists>processname

Waits for the specified process name to exist.

The system variable WFP_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WFP_TIMEOUT is used, WFP_RESULT will indicate whether or not the command ended successfully. If it timed out WFP_RESULT will be set to FALSE. If the process it was waiting for came into existance within the timeout setting, the WFP_RESULT value will be set to TRUE.

See also: ProcessExists 179, WaitProcessTerminated 182
Abbreviation: WPE

Example

```
WaitProcessExists>notepad.exe
MessageModal>Notepad now exists
```

4.16.28 WaitProcessTerminated

WaitProcessTerminated>processname

Waits for the specified process name to not exist.

The system variable WFP_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WFP_TIMEOUT is used, WFP_RESULT will indicate whether or not the command ended successfully. If it timed out WFP_RESULT will be set to FALSE. If the process it was waiting for terminated within the timeout setting, the WFP_RESULT value will be set to TRUE.

See also: ProcessExists 179, WaitProcessExists 182
Abbreviation: WPT

Example

```
WaitWindowClosed>notepad*
WaitProcessTerminated>notepad.exe
MessageModal>Notepad has left the building!
```

4.16.29 Wow64DisableRedirection

Wow64DisableRedirection

Disables 64 bit file system redirection.

On 64 bit versions of Windows the Windows\System32 folder is reserved for 64 bit applications. 32 bit versions of DLLs and applications are stored in Windows\SysWOW64. By default when a 32 bit process such as Macro Scheduler attempts to access a system resource it will be automatically redirected to the SysWOW64 folder to load the 32 bit resource. Even though you may explicitly declare "C:\Windows\System32" in a file operation the operating system will divert the call to the SysWow64 folder.

© 2015 MJT Net Ltd

Usually this is desirable (32 bit processes cannot load 64 bit DLLs) and there are 32 bit equivalents for almost all system functions.

However there may be occasions where a 32 bit equivalent does not exist and you explicitly want to execute a 64 bit system process. So you may want to disable 64 bit file system redirection. You can do this with the Wow64DisableRedirection command. **But use it carefully** and be sure to re-enable redirection immediately afterwards. Failing to do that could cause instability as it may prevent Macro Scheduler from loading a system DLL and therefore cause loss of functionality.

For more information refer to the Microsoft documentation at:
http://msdn.microsoft.com/en-us/library/aa384187(v=vs.85).aspx

A safer way to force access to the System32 folder is to refer to the SYS_NATIVE variable. E.g. the following will run the 64 bit version of Notepad:

Run>%SYS_NATIVE%\Notepad.exe

Using SYS_NATIVE is the safest way to bypass file system redirection.

See also: Wow64EnableRedirection 183

4.16.30 Wow64EnableRedirection

Wow64EnableRedirection

Re-enables 64 bit file system redirection.

See also: Wow64DisableRedirection 182

4.17 Mouse Commands

4.17.1 GetCaretPos

GetCaretPos>X,Y,Relative

GetCaretPos retrieves the X, Y coordinates of the text caret (text cursor) of the foreground window. The coordinates are returned in the variables provided. Set Relative to 0 to return absolute screen coordinates, or 1 to return coordinates relative to the window.

Abbreviation: GTP
See also: GetCursorPos 184

Example

```
SetFocus>notepad*
WaitReady>0
GetCaretPos>XCaret,YCaret,0
MouseMove>XCaret,YCaret
```

© 2015 MJT Net Ltd

4.17.2 GetCursorPos

GetCursorPos>X,Y[,cursorType]

Returns the X and Y coordinates of the current mouse cursor position. X and Y are variables in which to store the coordinates. The optional cursorType return variable can be used to return the cursor type.

Abbreviation : GCP
See also: GetWindowPos 245, GetCaretPos 183

Example

```
GetCursorPos>XCursor,YCursor
Message>Current Position : %XCursor%,%YCursor%
```

4.17.3 LClick

Simulates a left button mouse click at the current point on the screen.

The following commands will produce the same result :

LDown
LUp

Abbreviation : LCl
See also: LDown 185, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.4 LDblClick

Simulates a left mouse button double click at the current point on the screen.

The following will achieve the same result :

LClick
LClick

or

LDown
LUp
LDown
LUp

Abbreviation : LDb
See also: LDown 185, LUp 185, LClick 184, RClick 188, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

© 2015 MJT Net Ltd

4.17.5 LDown

Simulates a press of the left mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Issuing this command and then using MouseMove would implement dragging. Use LUp to complete the operation.

Abbreviation : LDo
See also: LClick 184, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.6 LUp

Releases the left mouse button. It is the latter half of a click.

See LDown.

See also: LDown 185, LClick 184, LDblClick 184, RClick 188, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.7 MClick

Simulates a middle button mouse click at the current point on the screen.

The following commands will produce the same result :

MDown
MUp

Abbreviation : MCl
See also: LDown 185, LUp 185, LDblClick 184, LClick 184, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.8 MDblClick

Simulates a middle mouse button double click at the current point on the screen.

The following will achieve the same result :

MClick
MClick

or

MDown
MUp
MDown
MUp

Abbreviation : MDb
See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp 188, MDown 186

© 2015 MJT Net Ltd

, MUp 187, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.9 MDown

Simulates a press of the middle mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Abbreviation : MDo
See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDblClick 188, RUp 188, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.10 MouseMove

MouseMove>X,Y

Moves the mouse cursor to screen position X,Y. 0,0 is the upper left hand corner of the screen. The maximum limits are determined by your screen resolution settings. Variables containing the coordinates can be used in the command.

To help determine a particular point on the screen, the macro window has a cursor monitor which updates as you move the cursor. See Creating Scripts 4.

Abbreviation : Mou
See also: MouseMoveRel 186, LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp 188, RDblClick 188, MouseOver 187

Example

If position 504,252 is within the area taken up by a button, the following script would cause that button to be clicked :

```
MouseMove>504,252
LClick
```

4.17.11 MouseMoveRel

MouseMoveRel>X,Y

Moves the mouse cursor to the position X,Y relative to the upper left corner of the window currently in focus. 0,0 will be the upper left hand corner of the active window. Variables containing the coordinates can be used in the command.

The advantage of this command over the MouseMove command, is that this will not fail when the window changes its position or resizes, or if the screen resolution is changed.

To help determine a particular point on the screen, the macro window has a cursor monitor which updates as you move the cursor. See Creating Scripts 4. To determine a point relative to a specific window, try moving that window so that its upper left corner is in the upper left corner of the screen (position 0,0). Maximising the app would achieve the same result. Then you can use the values displayed in the cursor position monitor of Macro Scheduler.

Abbreviation : MMR
See also: MouseMove 186, LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp

© 2015 MJT Net Ltd

188, RDblClick 188, MouseOver 187

Example

If position 40,50 is a point relative to the current window, on a button, the following script would cause that button to be clicked :

```
MouseMoveRel>40,50
LClick
```

4.17.12 MouseOver

MouseOver>window_title,button/object_caption

Attempts to position the mouse cursor over the specified button (or object) of the specified window.

window_title can contain an asterisk (*) as with all other window functions. For buttons that have a hot key associated with them, and represented on the button by an underscored letter, pass a & character before that letter. e.g.: for a button called 'Close', send &Close.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

window_title can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation : MVR

Example

```
RunProgram>rundll32.exe shell32.dll,Control_RunDLL TimeDate.cpl
WaitWindowOpen>Date and Time Properties
...
MouseOver>Date/Time*,OK
LClick
```

4.17.13 MUp

Releases the middle mouse button. It is the latter half of a click.

See MDown.

See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDblClick 188, RDown 188, MDown 186, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

© 2015 MJT Net Ltd

4.17.14 RClick

Simulates a right button mouse click at the current point on the screen.

The following commands will produce the same result :

RDown
RUp

Abbreviation : RCl
See also: LDown 185, LUp 185, LDblClick 184, LClick 184, RDown 188, RUp 188, RDblClick 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.15 RDblClick

Simulates a right mouse button double click at the current point on the screen.

The following will achieve the same result :

RClick
RClick

or

RDown
RLUp
RDown
RUp

Abbreviation : RDb
See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDown 188, RUp 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.16 RDown

Simulates a press of the right mouse button. This is like pressing the mouse button down but not releasing it. It is half of a click.

Abbreviation : RDo
See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDblClick 188, RUp 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

4.17.17 RUp

Releases the right mouse button. It is the latter half of a click.

See RDown.

See also: LClick 184, LDown 185, LUp 185, LDblClick 184, RClick 188, RDblClick 188, RDown 188, MDown 186, MUp 187, MDblClick 185, MClick 185, MouseMove 186, MouseMoveRel 186, MouseOver 187

© 2015 MJT Net Ltd

4.17.18 Toolbar

Toolbar>window_title,toolbar_index,button_index

NB: This function does not work in Windows 95

Positions the mouse cursor over the specified toolbar button of the specified window. Works only with toolbar objects of class ToolbarWindow32.

toolbar_index and button_index are integer values and begin at zero, where a toolbar_index of 0 indicates the first toolbar created on the window, and 1 the next and so on. Use Tools/View System Windows to find objects of class ToolbarWindow32 belonging to a window, and/or experiment to determine which index should be used.

Many modern menu systems are actually ToolbarWindow32 class objects rather than menu objects. Internet Explorer uses a ToolbarWindow32 for it's main menu.

window_title can contain an asterisk (*) as with all other window functions.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

Abbreviation : TBR

Example

```
Toolbar>Internet Explorer*,0,2
LClick
```

4.18 Numeric Functions

4.18.1 Add

Add>Value,Number

Adds Number to Value. Deprecated by Let 190 for numeric calculations.

Interpreted as Value = Value + Number

Value must be a variable containing either a numeric or date value. Number can be either a literal number or a variable containing a numeric value.

For date values this function will add the number of days, represented in Number to the given date value.

See also: Sub 191

© 2015 MJT Net Ltd

Example

```
Let>Counter=5
Add>Counter,2
```

i.e. Counter=Counter+2

In this example the numeric variable, Counter, is given a new value of 7.

4.18.2 Let

Let>variable_name=value

Let is used to assign a value to a variable.

Where parameters are passed to commands, variables can also be passed. Variables can also be embedded within a parameter by enclosing the variable name within % symbols.

Let can also be used to perform basic calculations, and to concatinate strings. NB. If you want to assign to a variable a string that already contains a + sign, add an extra + sign to avoid the two strings being concatinated.

The **value** can contain a complex expression:

Complex Expressions

The expression must be contained within curly braces "{" and "}". String literals must be delimited with double quotes ("), e.g.:
"string". Variables must be delimited with % symbols, e.g.: %VarA%.

Several types of operators and functions can be used with complex expressions. For more details see Complex Expressions 54.

Examples

```
Let>name=freddy
Let>a=5

Let>path=c:\Program Files\
Run Program>%path%myapp.exe

Let>k=k+1

Let>A=60-4

Let>A=5*3

Let>F=75/32

Let>Forename=John
Let>Surname=Smith
Let>Name=%Forename% %Surname%
```

Name would now equal John Smith

© 2015 MJT Net Ltd

```
Let>MyVal={Upper("WorLd")}

Let>MyVal={5 * 10 + 25}

Let>d={%a%+%b%*%c%-23}
```

4.18.3 SetRoundMode

SetRoundMode>roundingmode

Sets the rounding mode for the Round 55 function. Roundingmode can be one of:

0: Round to Nearest (Default)
1: Round Down
2: Round Up
3: Truncate

With the default rounding mode (nearest) if X is exactly halfway between two whole numbers, the result is always the even number.

4.18.4 Sub

Sub>Value,Number

Subtracts a number from a value. Deprecated by Let 190 for numeric calculations.

Interpreted as Value = Value - Number

Value must be a variable containing a numeric or date value. Number can be either a literal number or a variable containing a numeric value.

For date values this function will subtract the number of days, represented in Number from the given date value.

See also: Add 189, Let 190

Example

```
Let>Counter=5
Sub>Counter,2
```

i.e. Counter=Counter-2

In this example the numeric variable, Counter, is given a new value of 3.

```
Let>Counter=Counter-1
```

Will now do the same thing.

© 2015 MJT Net Ltd

4.19 Registry Functions

4.19.1 RegistryDelKey

RegistryDelKey>root_key,key

Not supported in Macro Scheduler Lite.

Removes a key from the registry.

To access the 64 bit registry hive set REG_64 to 1. Default is 0.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : RDK
See also: RegistryDelVal 192, RegistryReadKey 193, RegistryWriteKey 193, RegistryEnumKeys 192, RegistryEnumVals 193

Example

```
RegistryDelKey>HKEY_CURRENT_USER,Software\MJTNET\Temp
```

4.19.2 RegistryDelVal

RegistryDelKey>root_key,key,Value

Not supported in Macro Scheduler Lite.

Removes a value from the registry.

To access the 64 bit registry hive set REG_64 to 1. Default is 0.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : RDV
See also: RegistryDelKey 192, RegistryReadKey 193, RegistryWriteKey 193, RegistryEnumKeys 192, RegistryEnumVals 193

Example

```
RegistryDelVal>HKEY_CURRENT_USER,Software\MJTNET\Temp,TestVal1
```

4.19.3 RegistryEnumKeys

RegistryEnumKeys>root_key,key,result

Returns a list of keys beneath the specified registry key. The list is delimited by CRLF pairs.

© 2015 MJT Net Ltd

See also: RegistryDelKey 192, RegistryDelVal 192, RegistryReadKey 193, RegistryWriteKey 193, RegistryEnumVals 193

Example:

```
RegistryEnumKeys>HKEY_CURRENT_USER,Software\MJTNET,keys
MessageModal>keys
Separate>keys,CRLF,keysArr
```

4.19.4 RegistryEnumVals

RegistryEnumVals>root_key,key,result

Returns a list of value names beneath the specified registry key. The list is delimited by CRLF pairs.

See also: RegistryDelKey 192, RegistryDelVal 192, RegistryReadKey 193, RegistryWriteKey 193, RegistryEnumKeys 192

Example:

```
RegistryEnumVals>HKEY_CURRENT_USER,Software\MJTNET\MSched12,vals
MessageModal>vals
```

4.19.5 RegistryReadKey

RegistryReadKey>root_key,key,entry,result_variable

Not supported in Macro Scheduler Lite.

Reads the value of an entry from the registry. The value is stored in the given variable.

To access the 64 bit registry hive set REG_64 to 1. Default is 0.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : RRK
See also: RegistryDelKey 192, RegistryDelVal 192, RegistryWriteKey 193, RegistryEnumKeys 192, RegistryEnumVals 193

Example

```
RegistryReadKey>HKEY_CURRENT_USER,Control Panel\Colors,ActiveTitle,VActTitle
Message>VActTitle
```

4.19.6 RegistryWriteKey

Not supported in Macro Scheduler Lite.

RegistryWriteKey>root_key,key,entry,value

Creates or modifies a registry entry. If the key and entry do not exist they are created and the new

© 2015 MJT Net Ltd

value assigned to the entry. If the key and entry already exist, the value is changed to the value provided. The function will create integer entries if the value specified is an integer, or else the new value will be a string. To force an integer value to be written as a string set REG_INTASSTR to 1 before calling RegistryWriteKey.

To access the 64 bit registry hive set REG_64 to 1. Default is 0.

Use all registry functions with caution. If you are unfamiliar with the Windows Registry we recommend that you do not use these functions. Removing or modifying a registry entry that you did not create could cause your system to become unstable.

Abbreviation : RWK
See also: RegistryDelKey 192, RegistryDelVal 192, RegistryReadKey 193, RegistryEnumKeys 192, RegistryEnumVals 193

Example

```
RegistryWriteKey>HKEY_CURRENT_USER,MyStuff,MyName,Fred Bloggs
```

4.20 Running Programs/Files

4.20.1 ExecuteFile

ExecuteFile>file_to_execute[,parameters]

Executes a file using the application associated with the given file's filetype.

file_to_execute can include a full path.

Abbreviation : Exe
See also: RunProgram 194

Example

```
ExecuteFile>report.doc
```

or

```
Let>filename=c:\my documents\accounts.xls
ExecuteFile>filename
```

4.20.2 RunProgram

RunProgram>path

Executes a specified file. Files than can be executed are .exe, .bat, and .com files.

By setting the RP_WAIT variable to 1 prior to issuing the Run Program command the script will wait until the program launched by Run Program has terminated before continuing. The default value of RP_WAIT is 0.

Set RP_WAIT to 2 if you want the script to wait until it thinks the program is idle and ready for input.

© 2015 MJT Net Ltd

This only works for GUI applications and may not be appropriate for all applications. Some applications may appear ready for input before they really are. See WaitWindowOpen 185 and WaitReady 209 for other ways to wait until an application is ready.

By setting the RP_WINDOWMODE variable programs can be executed minimized, maximised, hidden or normal. RP_WINDOWMODE can be one of the following :

0: Hidden
1: Normal (default)
2: Minimized
3: Maximized

If you set RP_WINDOWMODE it is used by all subsequent Run Program commands, so remember to set it back if you don't want the same window mode to be used each time.

By default a message is displayed if this command encounters an error when running the specified program. Error messages from this command can be suppressed by setting RP_DISPLAYERROR to 0.

The result of the Run Program command is stored in the variable RP_RESULT.

The value of RP_RESULT differs depending on whether RP_WAIT is set to 0 or 1.

If RP_WAIT is 1 RP_RESULT will be set to the process exit code of the called program if successful. Therefore it will be zero if successful and the program does not return an exit code. -1 indicates that an error occured. If an error occurs the return code will be -1 followed by a colon followed by the error code. E.g. -1:2

If RP_WAIT is 0 (default) a value greater than 31 indicates success and the following values represent errors :

0 - The system was out of memory, or the executable file was corrupt, or relocations were invalid. (RP_WAIT=0 only)
2 - The file was not found.
3 - The path was not found.
5 - An attempt was made to dynamically link to a task, or there was a sharing or network protection error.
6 - The library required separate data segments for each task.
10 - The Windows version was incorrect.
11 - The executable file was invalid. It was either not a Windows-based application or there was an error in the .EXE image.
12 - The application was designed for OS/2.
13 - The application was designed for MS-DOS 4.0.
14 - The type of executable file was unknown.
15 - An attempt was made to load a real-mode application (developed for an earlier version of Windows).
16 - An attempt was made to load a second instance of an executable file containing multiple data segments that were not marked "read-only."
17 - Attempt in large-frame EMS mode to load a second instance of an application that links to certain non-shareable DLLs already in use.
18 - Attempt in real mode to load an application marked for protected mode only.

To run as admin, set RP_ADMIN to 1.

If attempting to run a 64 bit system process residing in the System32 folder on Windows x64 you can set RP_WIN64PROCESS to 1 to temporarily disable file system redirection 182. Alternatively refer to SYS_NATIVE instead of SYS_DIR.

Abbreviation : Run
See also: ExecuteFile 194

Example

© 2015 MJT Net Ltd

To open Notepad :

```
RunProgram>notepad.exe
```

A path may be specified if necessary :

```
RunProgram>c:\my programs\eudora\eudora.exe
```

To start notepad minimized :

```
Let>RP_WINDOWMODE=2
RunProgram>notepad.exe
```

4.21 Script Control

4.21.1 Sub Routines

4.21.1.1 Gosub

Gosub>Subroutine_Name[,value1[,value2[,...]]]

Branches to the specified subroutine (SRT). When that the subroutine returns, processing continues at the next line after the Gosub command.

Subroutines can be nested. Subroutines can be located anywhere within the script.

If values are passed on the Gosub line variables will be created to contain these values. The variables are named with the following format: SubroutineName_Var_1, Subroutine_Var_2, ... Subroutine_Var_n

See also SRT 197, END 198

Example

```
Gosub>EnterName
Gosub>CloseApp

//// Subroutines Below ////

SRT>FocusApp
   SetFocus>Data Entry Screen
End>FocusApp

SRT>EnterName
   Gosub>FocusApp
   Send>Firstname
   Press Tab
   Send>Surname
   Press Enter
End>EnterName

SRT>CloseApp
   Gosub>FocusApp
   Press ALT
```

© 2015 MJT Net Ltd

```
   Press F4
   Release ALT
End>CloseApp
```

4.21.1.2 SRT

SRT>Subroutine_Name

Identifies the start of a Subroutine block. Use End 198 to end the subroutine.

Subroutines can be nested. Subroutines can be located anywhere within the script.

In almost all circumstances subroutines should be allowed to end properly and it is best practice to do so. In other words execution should always continue on to the subroutine's End statement. While it is possible to jump out of a subroutine using a Goto statement the scope of execution will remain inside the subroutine unless and until execution returns to the subroutine and the subroutine's End statement is reached. This is most noticeable when using an OnEvent 204 event handler. If, for example, a KEY_DOWN event is set to fire a subroutine and code is allowed to jump out of the subroutine, OnEvent won't know that the subroutine has ended and therefore no subsequent KEY_DOWN events of the same type will fire. If you do need to cause execution to jump to a label after the subroutine has terminated use the SkipLabel 166 command.

By default variables have global scope. Variables can be set to have local scope by setting LOCALVARS to 1. See topic: Variable Scope 266.

See also Gosub 196, END 198, SkipLabel 166

Example

```
Gosub>EnterName
Gosub>CloseApp

//// Subroutines Below ////

SRT>FocusApp
   SetFocus>Data Entry Screen
End>FocusApp

SRT>EnterName
   Gosub>FocusApp
   Send>Firstname
   Press Tab
   Send>Surname
   Press Enter
End>EnterName

SRT>CloseApp
   Gosub>FocusApp
   Press ALT
   Press F4
   Release ALT
End>CloseApp
```

© 2015 MJT Net Ltd

4.21.1.3 END

END>Subroutine_Name

Identifies the end of a Subroutine block. Use SRT 197 to start the subroutine.

Subroutines can be nested. Subroutines can be located anywhere within the script.

See also Gosub 196, SRT 197

Example

```
Gosub>EnterName
Gosub>CloseApp

//// Subroutines Below ////

SRT>FocusApp
   SetFocus>Data Entry Screen
End>FocusApp

SRT>EnterName
   Gosub>FocusApp
   Send>Firstname
   Press Tab
   Send>Surname
   Press Enter
End>EnterName

SRT>CloseApp
   Gosub>FocusApp
   Press ALT
   Press F4
   Release ALT
End>CloseApp
```

4.21.1.4 SkipLabel

SkipLabel>LabelName

SkipLabel is used within a subroutine to tell the script to jump to a label *after* the subroutine has ended. It is bad practice to jump out of a subroutine using a Goto statement - as the subroutine may then never end, causing problems with, for example, OnEvent handlers. In some cases however it is useful to cause the script to jump to a label once the subroutine has finished. This is what SkipLabel does.

Note: SkipLabel is not supported by dialog event handler subroutines.

Example

```
Let>x=0
Repeat>x
  Let>x=x+1
  GoSub>ExampleSubroutine
Until>x=100

SRT>ExampleSubroutine
```

© 2015 MJT Net Ltd

```
  If>x=5
    SkipLabel>EndScript
  Endif
  //this bit still gets executed
  Let>y=x
END>ExampleSubroutine

Label>EndScript
MessageModal>y
```

4.21.2 Assigned

Assigned>Variable_Name,result

Used to determine whether or not a variable has been assigned. Assigned sets result to TRUE if Variable_Name exists, FALSE if not.

Abbreviation : ASS

Example

```
Assigned>strName,strNameExists
If>strNameExists=TRUE,DoName,SkipName
```

4.21.3 EndWhile

EndWhile

Use in conjunction with While to create a loop. The code between While and EndWhile will repeat while condition is true.

Note that the condition is evaluated at the start of the loop and therefore at the start of each iteration.

See also: While 167, Repeat 208

Example

```
Let>x=0
While>x<10
  Let>x=x+1
  MessageModal>x
EndWhile
```

4.21.4 Exit

Exit>[return_code]

Immediately terminates the current script. When used in a compiled macro you can set the executable's return code with return_code.

Example

© 2015 MJT Net Ltd

```
Exit>0
```

4.21.5 Goto

Goto>Label_Name

Causes execution to continue at the specified label, missing any commands in between. If the label does not exist an error message will be displayed.

Label_Name can be or include a variable name.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also: Label 202

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

4.21.6 If

If>expression[,true_label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

Evaluates **expression**. If the expression is true the first statements are executed. If the expression is false the statements after Else are executed. Else is optional. IF blocks can be nested. When label names are provided execution of the script jumps to the specified labels or subroutine names. If label names are specified Else and Endif are ignored and are unnecessary.

The **expression** can be *simple (legacy)*, or *complex*.

Complex Expressions

The expression must be contained within curly braces "{" and "}". String literals must be delimited with double quotes ("), e.g.:
"string". Variables must be delimited with % symbols, e.g.: %VarA%.

© 2015 MJT Net Ltd

Several types of operators and functions can be used with complex expressions. For more details see Complex Expressions 54.

Simple Expressions

Simple expressions can contain only two parts separated by one of the following operators:

= Equals
> Greater than
< Less than
<> Not Equal

Values in the expression can be numeric or string values, or variables containing such values. Spaces must not be used as simple expressions do not require string delimiters.

See also: IfNot 201, Label 202, Goto 200, IfFileChanged 124, IfFileExists 124, IfWindowOpen 151, Subroutines 197

Example

```
Let>a=5
//simple expression only
IF>a=5
    //do something
ELSE
    //do something else
ENDIF

//complex expression:
IF>{(%a% = 5) AND (%VarA% = "allen")}
   //do something
ELSE
   //do something else
ENDIF

//using labels
Label>Start
..
..
..
If>a<b,Start
```

4.21.7 IfNot

IfNot>expression[,true_label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

The reverse of If 200. Evaluates **expression**. If the expression is not true the first statements are executed. If the expression is true the statements after Else are executed. Else is optional. IF blocks can be nested. When label names are provided execution of the script jumps to the specified labels or subroutine names. If label names are specified Else and Endif are ignored and are unnecessary.

For more information on how to use this please see If 200.

© 2015 MJT Net Ltd

4.21.8 Include

Include>scriptfile

Includes code from an external script file. The code is executed and made available to the calling script. Therefore variables and subroutines in the external script file are made available to the calling script.

With Include the code in the Include file literally becomes part of the main script at run time. Including a script is rather like copying and pasting the code from it into the main script at runtime. To understand this better, step through a script that has an Include line in the debugger - you will see the code inserted into the script when the Include line is executed.

This differs markedly from the Macro 204 command which only runs the external script file as a separate entity, rather like running any other program.

See also: IncludeFromVar 202, Macro 204

4.21.9 IncludeFromVar

IncludeFromVar>variable_name

IncludeFromVar works like Include 202 but using code stored in a script variable rather than code from a file.

Any valid script code stored in a variable can therefore be executed as if it were part of the main script.

See also: Include 202

Example

```
LabelToVar>MyLabel,somevar
IncludeFromVar>somevar

/*
MyLabel:
Let>x=5
MessageModal>x
*/
```

4.21.10 Label

Label>Label_Name

Marks a point in the script to allow execution to be passed to that point by the Goto, and If.. commands.

Goto in conjunction with Label, can be used to create infinite loops. Use the If.. commands to cause conditional branching. To break out of infinite loops press Stop, or choose the Break option from the taskbar pop up menu.

See also: Goto 200, If 146, IfWindowOpen 151, IfFileExists 124, IfFileChanged 124

© 2015 MJT Net Ltd

Example

```
Label>Start
..
..
Goto>SecondBit
..
..
Label>SecondBit
..
```

4.21.11 Let

Let>variable_name=value

Let is used to assign a value to a variable.

Where parameters are passed to commands, variables can also be passed. Variables can also be embedded within a parameter by enclosing the variable name within % symbols.

Let can also be used to perform basic calculations, and to concatinate strings. NB. If you want to assign to a variable a string that already contains a + sign, add an extra + sign to avoid the two strings being concatinated.

The **value** can contain a complex expression:

Complex Expressions

The expression must be contained within curly braces "{" and "}". String literals must be delimited with double quotes ("), e.g.:
"string". Variables must be delimited with % symbols, e.g.: %VarA%.

Several types of operators and functions can be used with complex expressions. For more details see Complex Expressions 541.

Examples

```
Let>name=freddy
Let>a=5

Let>path=c:\Program Files\
Run Program>%path%myapp.exe

Let>k=k+1

Let>A=60-4

Let>A=5*3

Let>F=75/32

Let>Forename=John
Let>Surname=Smith
Let>Name=%Forename% %Surname%
```

© 2015 MJT Net Ltd

Name would now equal John Smith

```
Let>MyVal={Upper("WorLd")}

Let>MyVal={5 * 10 + 25}

Let>d={%a%+%b%*%c%-23}
```

4.21.12 Macro

Macro>file_name [/variable=value|variable [/variable=value|variable] ...]

Executes another script file. file_name must be a filename of a macro file. It is advisable to specify the full path should the path of the script file differ from the current path or change during the execution of the calling macro.

By default the extension of macro files is .scp, so for macros that are listed in Macro Scheduler add .scp to the macroname and prefix with the correct path. The path of a macro listed in Macro Scheduler is defined under Group Properties for the group the macro is listed in.

To pass values to the macro specify each one after a / character. The variable name given should exist in the script to be run. The value to assign to that variable is specified after the = character.

Data can be returned to the calling macro by setting the MACRO_RESULT variable in the called script. After the script has been run MACRO_RESULT is available to the calling script.

If the calling macro has a log file the script being called with the Macro command will log to the same log file. You can override this using the /LOGFILE parameter to set a log file for the sub macro.

Abbreviation : Mac

Examples

```
Macro>Defragment Disk.scp

Macro>%SCRIPT_DIR%\MyMoveFile.scp /source=c:\temp\myfile.bat /destination=c:\temp\myfile.bak
```

4.21.13 OnEvent

OnEvent>EventType,EventParm,ExtraParm,Subroutine

Not supported in Macro Scheduler Lite.

Establishes an event handler. When the event occurs the script branches to the specified Subroutine.

Event Types:

EventType	EventParm	ExtraParm	Description
WINDOW_OPEN	Window_Title	WF_TYPE	Triggers when specified window is open. See IfWindowOpen 247 for window title & WF_TYPE syntax.
WINDOW_NO	Window_Title	WF_TYPE	Triggers when specified window is closed.

© 2015 MJT Net Ltd

TOPEN			See IfWindowOpen 247 for window title & WF_TYPE syntax.
WINDOW_NEWACTIVE	0	0	Triggers when the foreground window changes.
FILE_EXISTS	Filename	0	Triggers when the specified file exists.
FILE_NOTEXISTS	Filename	0	Triggers when the specified file does not exist.
KEY_DOWN	Key/KeyCode	Modifier	Triggers when the key is down. See WaitKeyDown 163
DIALOG_EVENT	Dialog Name	Modal Result	Triggers when dialog's modal result matches e.g. when a button is pressed.
DIALOG_CHANGE	Dialog Name	Object Name	Triggers when object value is changed.
PROCESS_EXISTS	Process Name		Triggers when specified process exists.
PROCESS_NOTEXISTS	Process Name		Triggers when specified process does not exist.
PIXEL_COLOR	x:y	color_code	Triggers when color at specfied x,y position matches.
DATE	Date in YYYYMMDD format		Triggers when current date matches given date.
TIME	Time in HHMM format		Triggers when current time matches given time
CUSTOM	SubroutineName	VariableName	Custom Trigger - runs given subroutine periodically and triggers when specified variable = TRUE

Explanations:

WINDOW_OPEN and WINDOW_NOTOPEN
EventParm takes a window title (or handle if WIN_USEHANDLE has already been set to 1).
ExtraParm takes a WF_TYPE value. See SetFocus 249 and IfWindowOpen 247 for explanation.

By definition these event types cause the script to continually enumerate and loop through the list of windows open on the system. This is an intensive process and you may therefore see CPU usage increase. For most people this is not a problem. However, if you wish to reduce CPU usage you can set WIN_SLEEP to 1 at the expense of processing time (in theory setting WIN_SLEEP to 1 will slow the checks down while reducing CPU usage).

WINDOW_NEWACTIVE
EventParm and ExtraParm are not used. Set to zero.

FILE_EXISTS and FILE_NOTEXISTS
EventParm takes a filename.
ExtraParm is not used. Set to zero.

KEY_DOWN
EventParm takes a character for an ordinary key. For other keys use the virtual key code preceeded by VK. See WaitKeyDown 163 for more information.
ExtraParm is used to specify a modifier key:

0: No modifier key
1: SHIFT key must also be pressed
2: CONTROL key must also be pressed
3: ALT key must also be pressed
4: SHIFT + ALT keys must be pressed
5: CONTROL + ALT keys must be pressed
6 SHIFT + CONTROL keys must be pressed
7: CONTROL + ALT + SHIFT keys must also be pressed

© 2015 MJT Net Ltd

8: Windows key must also be pressed

DIALOG_EVENT
EventParm is the dialog name. ExtraParm is a modal result value to detect. This could be the modal result of a button or menu option. So when that button is pressed the subroutine is triggered.

DIALOG_CHANGE
EventParm is the dialog name. ExtraParm is the object name. When the value of the object is changed by the user the subroutine is triggered. Valid object types include Edit, Memo, ComboBox, ListBox, RadioGroup and CheckBox.

PIXEL_COLOR
EventParm should have the format X:Y where X and Y are the x and y positions of the screen coordinate and should be separated by a colon. ExtraParm is the color code to watch for. Use the cursor monitor in the editor to determine the desired color code.

DATE
EventParm takes a date in YYYYMMDD format.

TIME
EventParm takes an hour in HHMM format (24 hour format).

CUSTOM
Allows to create any kind of event trigger. EventParm is a subroutine which will be executed continually. ExtraParm is a variable name. When the variable name becomes TRUE the trigger is fired. See example below which creates a trigger based on a registry value.

To disable an event handler issue the same OnEvent code but with the subroutine omitted (i.e. an empty string).

Abbreviation: ONE

Example

```
OnEvent>WINDOW_OPEN,Notepad*,2,DoNotepad
OnEvent>WINDOW_NOTOPEN,Notepad*,2,DoNotepadNotOpen
OnEvent>WINDOW_NEWACTIVE,0,0,DoNewWindow
OnEvent>KEY_DOWN,VK32,3,KeyPress
```

© 2015 MJT Net Ltd

```
OnEvent>CUSTOM,MyTriggerSub,DoIT,DoSomething

Label>start
  Wait>1
  If>NotepadOpen=1
    Message>Notepad is open
  Else
    Message>Notepad is not open
  Endif
Goto>start

SRT>DoNotepad
  Let>NotepadOpen=1
END>DoNotepad

SRT>DoNewWindow
  GetActiveWindow>title,x,y
  MessageModal>New window: %title%
END>DoNewWindow

SRT>DoNotepadNotOpen
  Let>NotepadOpen=0
END>DoNotepadNotOpen

SRT>KeyPress
  MessageModal>ALT+Space was pressed
END>KeyPress

SRT>MyTriggerSub
  //This custom trigger monitors a registry key value
  RegistryReadKey>HKEY_CURRENT_USER,Software\MySoft,MyValue,res
  If>res=FIRE
    Let>DoIT=TRUE
  Endif
END>MyTriggerSub

SRT>DoSomething
  //reset the registry key and do something
  RegistryWriteKey>HKEY_CURRENT_USER,Software\MySoft,MyValue,IDLE
  //do something else
  Let>DoIT=FALSE
END>DoSomething
```

To disable an event handler call OnEvent again with the same parameters but instead of the subroutine name issue an empty string. E.g.:

```
//enable
OnEvent>KEY_DOWN,VK32,0,KeyPress

...
...

//disable it
OnEvent>KEY_DOWN,VK32,0,

..
..

//enable it again
OnEvent>KEY_DOWN,VK32,0,KeyPress
```

© 2015 MJT Net Ltd

4.21.14 Remark

Remark>Some Comment

The remark statement is ignored by the interpreter. It exists simply to allow comments to be placed in the code. You can also use // E.g.:

// This is a comment

Comments must be on a line of their own.

Several lines of code can also be commented out in one go by using /* before and */ after the lines to be commented. E.g.:

```
/*
Let>r=5
Repeat>r
  DDERequest>Excel,efile.xls,R%r%C2,Name,60
 Let>r=r+1
Until>r=5
*/
```

You can also use /* ... */ at the end of lines of real code

4.21.15 Repeat

Repeat>variable

Use in conjunction with Until. Iterates the code from the Repeat statement to the Until statement until the specified expression in the Until statement becomes true. Until can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Alternatively Until can take a full complex expression 54 (specified between curly braces).

Until will loop back to the last Repeat that has the same counter variable.

See also: Until 209

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
```

© 2015 MJT Net Ltd

4.21.16 Until

Until>variable,finalvalue

Use in conjunction with Repeat. Iterates the code from the Repeat statement to the Until statement until the specified expression in the Until statement becomes true. Until can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Until will loop back to the last Repeat that has the same counter variable.

See also: Repeat 208

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%

Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
```

4.21.17 Wait

Wait>seconds_to_wait

This command makes Macro Scheduler pause for the specified number of seconds.

See also: WaitWindowOpen 253, WaitWindowClosed 251, WaitPixelColor 156, WaitCursorChanged 181

Example

To wait 5 seconds :

```
Wait>5
```

This will work too :

```
Let>WaitTime=5
Wait>WaitTime
```

4.21.18 WaitReady

WaitReady>paint_events

WaitReady suspends script execution until the foreground window has finished processing mouse,

© 2015 MJT Net Ltd

keyboard, show window, and optionally, paint events. Issue 1 to include paint events, and 0 to exclude paint events. This command can therefore be used to wait until the active application is ready to receive keyboard and mouse events in most situations.

Abbreviation : WRD
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitCursorChanged 181, WaitPixelColor 156

Example

```
Run Program>"C:\Program Files\Microsoft Office\Office\WINWORD.EXE"
WaitWindowOpen>Microsoft Word*
WaitReady>0
Message>Word is now ready for input
```

4.21.19 While

While>condition

Use in conjunction with EndWhile to create a loop. The code between While and EndWhile will repeat while condition is true.

Note that the condition is evaluated at the start of the loop and therefore at the start of each iteration.

While can take one of the following simple expressions:

variable,value	for backward compatibility. Same as variable=value
variable=value	Until variable equals the value specified
variable>value	Until variable is greater than the value specified
variable<value	Until variable is less than the value specified
variable<>value	Until variable does not equal the value specified

Alternatively While can take a full complex expression 54 (specified between curly braces).

See also: EndWhile 163, Repeat 208

Example

```
Let>x=0
While>x<10
  Let>x=x+1
  MessageModal>x
EndWhile
```

4.22 String Handling

4.22.1 AESEncrypt

AESEncrypt>source,SHA256Password,ENCRYPT|DECRYPT,target

Not supported in Macro Scheduler Lite.

© 2015 MJT Net Ltd

Uses AES encryption to encrypt or decrypt source to target using the specified password which is SHA256 hashed by the function (the password does not need hashing prior to use as it is hashed internally).

In Macro Scheduler 14.0.14 AES was reimplemented to address some Unicode issues. If this causes compatibility issues with data encrypted using the old method you can switch back to using the old algorithm by setting AES_LEGACY to 1.

See also: Hash 215

Example

```
//Create a password
Let>mypassword=this is a secret

Let>strText=the quick brown fox jumped over the lazy dog

//encrypt the string with AES
AESEncrypt>strText,mypassword,ENCRYPT,encrypted_data

//as encrypted data is binary use Base64 to encode it to a string
Base64>encrypted_data,ENCODE,encoded_encrypted_data

..
..

//decode and decrypt
Base64>encoded_encrypted_data,DECODE,encrypted_data
AESEncrypt>encrypted_data,mypassword,DECRYPT,strText2
```

4.22.2 Base64

Base64>source,ENCODE|DECODE,target

Encodes or decodes source to target using Base64. Base64 encodes data using pure ASCII characters. It is therefore ideal for storing binary data in text files etc and is commonly used for sending binary attachments in email.

By default Base64 works with ANSI strings only. To enable encoding/decoding of Unicode strings set BASE64_UNICODE to 1 first.

This function is most useful when used with the Crypt 212 function. Crypt results in binary data, so Base64 can be used to encode that binary data to a plain text string which can then safely be stored in a text file or database.

Abbreviation: Bas
See also: Crypt 212

Example

```
Crypt>abc,hello,cryptval
Base64>cryptval,ENCODE,ascii_cryptval
WriteLn>d:\test.txt,r,ascii_cryptval
```

© 2015 MJT Net Ltd

```
..

ReadLn>d:\test.txt,1,ascii_cryptval
Base64>ascii_cryptval,DECODE,bin_cryptval
Crypt>abc,bin_cryptval,clean
```

4.22.3 ConCat

ConCat>string1,string2

Concatenates string1 with string2. String1 must be a variable containing a string. String2 can be a literal string or a variable. The result is that string1 has string2 appended to it.

Abbreviation : Con

Example

```
Let>path=c:\temp\
ConCat>path,myfile.txt
```

In this example path becomes 'c:\temp\myfile.txt'

4.22.4 Crypt

Crypt>key,source,target

Encrypts or decrypts source to target using specified key. Encrypts a plain ANSI string, decrypts an encrypted string if the same key as used for the encryption is specified.

By default the encryption strength is set to 32 bit. This can be changed to medium level 64 bit or high level 96 bit using the CRYPT_LEVEL variable:

1 = Default 32 bit
2 = Medium 64 bit
3 = High 96 bit

Abbreviation: Cry
See also: Base64 211

Example

```
Crypt>abc,hello,cryptval

..

Crypt>abc,cryptval,clean
```

Crypt often produces binary output. Therefore, if writing encrypted strings to text files first encode the string with Base64. Decode the string when reading from the file before decrypting. E.g.:

```
//Encrypt the text
Crypt>abc,texttoencrypt,encdata
```

© 2015 MJT Net Ltd

```
//Encode with Base64
Base64>encdata,ENCODE,encdata

//Write to the file
WriteLn>file,r,encdata

//To read the data back and decrypt:
ReadLn>file,1,encdata
//UnEscape the encoded data
Base64>encdata,DECODE,encdata
//Decrypt
Crypt>abc,encdata,decryptedtext
```

4.22.5 ExtractFileExt

ExtractFileExt>filename,result

Extracts the extension (including the dot) from the given filename and returns it in result.

See also: ExtractFleName 213, ExtractFilePath 213
Abbreviation: EFE

Example:

```
Let>filename=c:\users\documents\somefile.doc

//ext_only will become ".doc"
ExtractFileExt>filename,ext_only
```

4.22.6 ExtractFileName

ExtractFileName>filename,result

Extracts the name and extension parts from the given filename and returns it in result.

See also: ExtractFleExt 213, ExtractFilePath 213
Abbreviation: EFN

Example:

```
Let>filename=c:\users\documents\somefile.doc

//name_only will contain "somefile.doc"
ExtractFileName>filename,name_only
```

4.22.7 ExtractFilePath

ExtractFilePath>filename,result

Extracts the directory path from the given filename and returns it in result.

See also: ExtractFileExt 213, ExtractFileName 213
Abbreviation: EFP

© 2015 MJT Net Ltd

Example:

```
Let>filename=c:\users\documents\somefile.doc

//path_only will contain "c:\users\documents"
ExtractFilePath>filename,path_only
```

4.22.8 Format

Format>format_string,data,result

Formats the given data into the string provided.

Each data formatting substring starts with a % and ends with a data type indicator :

d = Decimal (integer)
e = Scientific
f = Fixed
g = General
m = Money
n = Number (floating)
p = Pointer
s = String
u = Unsigned decimal
x = Hexadecimal

The general format of each formatting substring is as follows:

%[-][Width][.Precision]Type

Where items in square brackets are optional.

See examples below.

Example

```
//just insert a string in a string
Let>data=45.12
Format>string is: %s,data,result
//Result becomes: "string is: 45.12"

//set floating point number to 6 decimal places
Format>%.6n,data,result
//Result becomes: 45.120000

Let>data=1234
Format>%.8d,data,result
//Result becomes: 00001234

//left padded:
Format>Padded: <%7d>,data,result
//Result becomes: "Padded: <   1234>"

//Right padded:
Format>Padded: <%-7d>,data,result
//Result becomes: "Padded: <1234   >"
```

© 2015 MJT Net Ltd

```
//Money
Let>data=134.647
Format>Price: %m,data,result
//Result becomes: "Price: $134.65" (currency symbol from system)

//To hex
Format>%x,324234,result
//Result becomes: 4F28A
```

4.22.9 Hash

Hash>string,algorithm,result

Not supported in Macro Scheduler Lite.

Returns the hash of the given string using the specified algorithm

Supported algorithms are:

MD5
SHA1
SHA256

Example

```
Let>str=the quick brown fox jumped over the lazy dog
Hash>str,MD5,md5Hash
```

4.22.10 HTMLDecode

HTMLDecode>text,result

Replaces the HTML character entities with the corresponding HTML special characters.

See also: HTMLEncode 215

4.22.11 HTMLEncode

HTMLEncode>text,result

Replaces the characters with special HTML significance with the corresponding HTML character entities.

See also: HTMLDecode 215

4.22.12 JSONParse

JSONParse>json_string,json_path,result

Uses JSONPath syntax to extract data from a JSON string.

© 2015 MJT Net Ltd

Example

```
HTTPRequest>http://ip.jsontest.com/,,GET,,JSON
JSONParse>JSON,ip,myIP
MessageModal>Your pubic IP is: %myIP%
```

Example

```
/*
MyJSON:
{ "uid" : "1234",
  "clients" : ["client1","client2","client3"],
  "people" : [{"Name":"Marcus","Age":"21"},{"Name":"Dorian","Age":"18"}],
  "color" : "red",
  "size" : 14 }
*/

LabelToVar>MyJSON,sJSON

JSONParse>sJSON,uid,result
JSONParse>sJSON,clients,result
JSONParse>sJSON,clients[1],result
JSONParse>sJSON,people[1].Name,result
```

4.22.13 LabelToVar

LabelToVar>LabelName,VariableName[,WantLineBreaks,IgnoreVariables,EndToken]

Reads the contents of a data label into the specified variable.

The optional parameter WantLineBreaks can be set to 0 to force LabelToVar to discard line breaks. By default line breaks are retained (WantLineBreaks=1). Ommitting WantLineBreaks or setting it to anything other than zero will cause line breaks to be included as default. Setting to zero causes line breaks to be removed.

Optional parameter IgnoreVariables can be set to 1 to ignore variables inside % symbols in the string. The default is 0 (all variables are resolved).

Optional parameter EndToken allows you to set what can be used to signify the end of the block. The default is the end comment token ("*/"). Whatever is specified must exist at the start of the line.

Note that WantLineBreaks, IgnoreVariables and EndToken must all be specified if any one is required.

A data label takes the same format as used by the binary import tool which imports binary data into a label.

LabelToVar is useful where you need to specify long constant strings in your script and wish to avoid concatenating each line.

Example:

```
LabelToVar>SQL1,mySQL

/*
SQL1:
```

© 2015 MJT Net Ltd

```
select id, forename, surname
from customers
where
surname = '%strSurname%'
*/
```

4.22.14 Length

Length>string,result

Returns the length of the given string

Abbreviation : Len
See also: MidStr 217, Position 218, ConCat 212

Example

```
Length>Hello World,Len
```

4.22.15 LowerCase

LowerCase>string,result

Returns in result the lowercase version of string.

See also: UpperCase 220
Abbreviation: LOW

4.22.16 LTrim

LTrim>string,result

Returns in result the string with leading spaces and control characters removed

See also: Trim 220, RTrim 219
Abbreviation: LTR

4.22.17 MidStr

MidStr>string,start,length,result

Returns a substring of specified length from a given position in a string. result is a variable in which to store the returned string. Any parameter can be a variable containing the appropriate values.

Abbreviation : Mid
See also: Position 218, ConCat 212, Length 217

Example

© 2015 MJT Net Ltd

In the following example, the variable somevalue becomes equal to 'Happy' :

```
MidStr>Happy Birthday,1,5,somevalue
Message>somevalue
```

4.22.18 Position

Position>substring,string,start,result[,relative]

Returns the starting position of a substring in a string. The search commences at the position specified in start. If found the starting position of the substring is returned in the result variable. If no match is found this value will be zero.

If string is an empty string (length zero) or start is greater than the length of the string then the result will be zero.

The optional parameter 'relative' can be used to determine whether the position returned is relative to the start position or an absolute position in the string being searched. The default is TRUE (compatible with previous versions). Set to FALSE to return the absolute position.

Abbreviation : Pos
See also: Length 217, MidStr 217, ConCat 212

Example

In this example, StartPos will contain the value 4 :

```
Position>Smith,Mr Smith,1,StartPos
```

4.22.19 RegEx

RegEx>pattern,text,easypatterns,matches_array,num_matches,replace_flag[,replace_string,replace_result]

Performs the regular expression in *pattern* on *text*, returning the number of sub matches in *num_matches* and the text found in *matches_array*. RegEx is compatible with the Perl 5.10 regular expression syntax using the PCRE library.

matches_array is the name of a variable to store the results. The first match is stored in matches_array_1, the second in matches_array_2, etc.

To perform a replacement set *replace_flag* to 1 and specify the replacement string in *replace_string* and the return variable in *replace_result*. Otherwise set *replace_flag* to 0.

By setting *easypatterns* to 1 *pattern* can use EasyPatterns syntax. EasyPatterns uses an english-like structure to simplify the use of regular expressions. See the EasyPatterns Reference.

Abbreviation: RGX

Examples

```
//Find an IP address:
Let>text=This is an IP address: 192.168.10.42
```

© 2015 MJT Net Ltd

```
Let>pattern=(\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b)
RegEx>pattern,text,0,matches,num,0
MessageModal>matches_1

//Find an IP address with EasyPatterns:
Let>text=This is an IP address: 192.168.10.42
Let>pattern=[IPAddress]
RegEx>pattern,text,1,matches,num,0
MessageModal>matches_1

//Replacement:
Let>text=My Email Address: freddy@mjtnet.com
Let>pattern=[EmailAddress]
RegEx>pattern,text,1,matches,num,1,new_address@server.com,text
MessageModal>text
```

4.22.20 RTrim

RTrim>string,result

Returns in result the string with trailing spaces and control characters removed

See also: Trim 220, LTrim 217
Abbreviation: RTR

4.22.21 Separate

Separate>list,delimiter,returnvar

Separate takes a list, a delimiter and returns the elements of the list. The command returns a number of variables, one for each list element, each with the suffix "_n" where n is the index of the element in the list. The variable names are determined by the name given in returnvar. e.g. returnvar_1, returnvar_2, etc. Also returned is the number of elements in the list, in returnvar_count.

Abbreviation : SEP

Example

```
GetFileList>c:\temp\*.*,files
Separate>files,;,file_names
MessageModal>Num Files: %file_names_count%
If>file_names_count=0,end
Let>k=0
Repeat>k
  Let>k=k+1
  Message>file_names_%k%
Until>k,file_names_count
Label>end
```

4.22.22 StringReplace

StringReplace>sourcestring,find,replace,newstring

Creates a new string, newstring, by searching sourcestring for all ocurrences of find, replacing them with replace.

© 2015 MJT Net Ltd

Abbreviation: RPL

See also: RegEx 218, Position 218, MidStr 217

Examples

```
Let>string=Your name is "Fred"
StringReplace>string,",",vbEscapedString

StringReplace>Good evening,evening,morning,newgreeting
```

4.22.23 Trim

Trim>string,result

Returns in result the string with leading and trailing spaces and control characters removed

See also: LTrim 217, RTrim 219
Abbreviation: TRI

4.22.24 UpperCase

UpperCase>string,result

Returns in result the uppercase version of string.

See also: LowerCase 217
Abbreviation: UPP

4.22.25 XMLParse

XMLParse>XML,XMLPath,result_string,num_items

Extracts data from an XML string using XMLPath syntax. If returning an individual item the result will be in *result_string*. If the XMLPath refers to a collection the number of items will be returned in *num_items*. See example below.

Example

© 2015 MJT Net Ltd

```
LabelToVar>XML,sXML
XMLParse>sXML,/bookstore/book,val,numBooks
Let>k=0
Repeat>k
  Let>k=k+1
  XMLParse>sXML,/bookstore/book[%k%]/title/text(),val,len
  MessageModal>val
Until>k=numBooks

/*
XML:
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
*/
```

4.23 Text Capture

© 2015 MJT Net Ltd

4.23.1 GetControlText

GetControlText>WindowTitle,ClassName,Instance,Result

Returns the text of the control specified by ClassName and Instance on the window specified by WindowTitle.

Result contains the text of the control if found. It the window is not found Result contains ##NOSUCHWINDOW##. If the class and instance was not found ##NOSUCHOBJECT## is returned.

If WIN_USEHANDLE is set to 1 WindowTitle must be a window handle.

To determine class names of objects on windows use View System Windows 36. A window may contain several objects of the same class name. Instance is used to determine which instance of the class to use.
GetControlText retrieves the published text property of the specified object. Not all text that you see on the screen is retrievable in this way. Some text is painted via lower level routines and some text is graphical. Some objects, such as labels, are not windowed controls, and therefore text associated with them cannot be retrieved with GetControlText. Try the new GetTextAtPoint, GetTextInRect and GetWindowTextEx commands which use lower level hooks to trap more text.

Abbreviation: GCT
See also: SetControlText 261, GetTextAtPoint 222, GetTextInRect 223, GetWindowTextEx 225, WaitScreenText 225, GetTextPos 224

Example:

```
GetControlText>Notepad*,Edit,1,npEdit
```

npEdit will now contain the contents of the notepad edit window.

4.23.2 GetTextAtPoint

GetTextAtPoint>x,y,result_variable,char_pos

Not supported in Macro Scheduler Lite.

Retrieves the text at the specified screen position. Any text retrieved is returned in result_variable. char_pos returns the zero-based index in result_variable of the character at position x,y.

GetTextInit 223 must be called before this function will work.

NB: This function requires Windows 2000 or above. It has no effect in Windows 95/98/NT.

Abbreviation: GTA
See also: GetTextInRect 223, GetWindowTextEx 225, WaitScreenText 225, GetTextInit 223, GetTextReset 224, GetTextPos 224

Example:

```
GetTextInit
```

© 2015 MJT Net Ltd

```
GetTextAtPoint>200,400,strText,nChar
if>nChar>-1
  MidStr>strText,nChar,1,sChar
Endif
MessageModal>%strText% (%sChar%)
```

4.23.3 GetTextInit

GetTextInit

Not supported in Macro Scheduler Lite.

Loads the text capture hooks. Must be called before any of the following functions will work:

GetTextAtPoint 222
GetTextInRect 223
GetTextPos 224
GetWindowTextEx 225
WaitScreenText 225

Failing to call GetTextInit before calling any of the above may result in an error.

Note that prior to version 13 this command was not necessary. In v13 and above all macros that make use of any of the above commands should have GetTextInit added beforehand.

4.23.4 GetTextInRect

GetTextInRect>left,top,right,bottom,result_variable

Not supported in Macro Scheduler Lite.

Retrieves text found in the screen rectangle bound by coordinates left,top and right,bottom.

GetTextInit 223 must be called before this function will work.

NB: This function requires Windows 2000 or above. It has no effect in Windows 95/98/NT.

Abbreviation: GTI
See also: GetTextAtPoint 222, GetWindowTextEx 225, WaitScreenText 225, GetTextReset 224, GetTextInit 223, GetTextPos 224

Example:

```
GetTextInit
GetTextInRect>0,0,200,400,strText
MessageModal>strText
```

© 2015 MJT Net Ltd

4.23.5 GetTextPos

GetTextPos>text_to_find,left,top,right,bottom

Searches the screen for the specified text and if found returns the containing rectangle's (left,top,right,bottom) screen coordinates.

If the text is not found left, top, right and bottom will all be set to -1.

The text can contain regular expressions.

GetTextInit 223 must be called before this function will work.

See also: GetTextAtPoint 222, GetWindowTextEx 225, WaitScreenText 225, GetTextReset 224, GetTextInit 223, GetTextInRect 223

Example:

```
GetTextInit
GetTextPos>File,X1,Y1,X2,Y2

//click in center
Let>x={%X1%+((%X2%-%X1%) div 2)}
Let>y={%Y1%+((%Y2%-%Y1%) div 2)}
MouseMove>x,y
LClick
```

4.23.6 GetTextReset

GetTextReset

Not supported in Macro Scheduler Lite.

Does the same as GetTextInit

4.23.7 GetWindowText

GetWindowText>window_title,text

GetWindowText retrieves all the detectable text contained within the specified window. Specify the window using the exact window title. The text is retrieved as a list with each object's text on a new line.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

GetWindowText retrieves the published text property of the specified window/object. Not all text that you see on the screen is retrievable in this way. Some text is painted via lower level routines and some text is graphical. Some objects, such as labels, are not windowed controls, and therefore text associated with them cannot be retrieved with GetWindowText. Try the new GetTextAtPoint, GetTextInRect and GetWindowTextEx 225 commands which use lower level hooks to trap more text.

Abbreviation : GWT
See also: GetWindowTextEx 225, GetTextAtPoint 222, GetTextInRect 223

© 2015 MJT Net Ltd

Example

```
GetWindowText>Document - WordPad,WordPadText
```

4.23.8 GetWindowTextEx

GetWindowTextEx>window_title,text

Not supported in Macro Scheduler Lite.

Retrieves visible text in the specified window. Unlike GetWindowText which works by accessing each object's published text property GetWindowTextEx uses lower level hooks to detect all text written to the screen and therefore is capable of capturing far more text.

The specified window is momentarily focused when the function is called.

GetTextInit 223 must be called before this function will work.

NB: This function requires Windows 2000 or above. It has no effect in Windows 95/98/NT.

Abbreviation: GWE
See also: GetTextAtPoint 222, GetTextInRect 223, WaitScreenText 225, GetTextInit 223, GetTextReset 224

Example:

```
GetTextInit
GetWindowTextEx>Document - WordPad,WordPadText
```

4.23.9 WaitScreenText

WaitScreenText>text

Waits for the specified text to appear on the screen.

The system variable WST_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If the command completes within the timeout period the value WST_RESULT will be set to TRUE. Otherwise WST_RESULT will be false.

NB: This function requires Windows 2000, XP, Server 2003, or Vista. It has no effect in Windows 95/98/NT.

Abbreviation: WST
See also: GetTextAtPoint 222, GetTextInRect 223, GetWindowTextEx 225, GetTextReset 224

Example:

```
WaitScreenText>Done
MessageModal>Complete
```

© 2015 MJT Net Ltd

4.24 VBScript Commands

4.24.1 VBEND

VBEND

Not supported in Macro Scheduler Lite.

Marks the end of a block of VBScript code. Macro Scheduler reads from VBSTART to VBEND and stores the VBScript code between the two markers for later execution by the VBRun, or VBEval commands.

If you wish to evaluate VBScript expressions in your scripts you will need a VBSTART/VBEND block at the start of your script even if it contains no code.

See also: VBEval 226, VBRun 228, VBSTART 228
Links to VBScript Documentation: http://www.mjtnet.com/resources.htm

Example

```
VBSTART

Function MultiplyNums (d,a)
  MultiplyNums = d * a
End Function

Sub DisplayMessage (msg)
  MsgBox msg
End Sub

Function GetName
  GetName = InputBox("Enter Your Name : ")
End Function

VBEND

Let>a=5
VBEval>MultiplyNums(%a%,2),answer
MessageModal>answer

VBRun>DisplayMessage,Hello World

VBEval>GetName,name
MessageModal>Your Name is : %name%
```

4.24.2 VBEval

VBEval>Function([parms]),result

Not supported in Macro Scheduler Lite.

Evaluates a VBScript expression, or function in the proceeding VBScript code block. Parameters can be passed to the function. The result of the function or expression is stored in the result variable

© 2015 MJT Net Ltd

which is a regular Macro Scheduler variable.

This command sometimes causes confusion. As it is evaluating a VBScript expression the syntax used in the first part of the command - the VBScript expression - should be valid VBScript syntax. To pass Macro Scheduler variables, embed them with the % symbol. If passing a Macro Scheduler variable as a string, remember that VBScript expects strings with quote marks around them. See examples below.

Note that since VBEval evaluates any valid VBScript expression it can do more than just execute VBScript functions. E.g. It could be used to return the value of a VBScript variable, or any other expression, such as a mathematical calculation.

It is possible to set the VBScript code timeout using the VBS_TIMEOUT variable. By default VBScript code will never timeout. To set a timeout, set VBS_TIMEOUT to a value in milliseconds.

Abbreviation : VBE
See also: VBRun 228
Links to VBScript Documentation: http://www.mjtnet.com/resources.htm

Example

```
VBSTART

Function MultiplyNums (d,a)
  MultiplyNums = d * a
End Function

Sub DisplayMessage (msg)
  MsgBox msg
End Sub

Function GetName
  GetName = InputBox("Enter Your Name : ")
End Function

Function PointlessStringExample(somestring)
  DisplayMessage(somestring)
  StringExample = somestring
End Function

VBEND

Let>a=5
VBEval>MultiplyNums(%a%,2),answer
MessageModal>answer

VBRun>DisplayMessage,Hello World

VBEval>GetName,name
MessageModal>Your Name is : %name%

Let>text=Hello World
VBEval>PointlessStringExample("%text%"),sametext

//Evaluate built in VBScript code,
//no code in VBSTART/VBEND block required
VBEval>Timer,elapsed
VBEval>MsgBox(%elapsed%),nul
```

© 2015 MJT Net Ltd

4.24.3 VBSTART

VBSTART

Not supported in Macro Scheduler Lite.

Marks the start of a block of VBScript code. Macro Scheduler reads from VBSTART to VBEND and stores the VBScript code between the two markers for later execution by the VBRun, or VBEval commands.

If you wish to evaluate VBScript expressions in your scripts you will need a VBSTART/VBEND block at the start of your script even if it contains no code.

See also: VBEval 226, VBRun 228, VBEND 226
Links to VBScript Documentation: http://www.mjtnet.com/resources.htm

Example

```
VBSTART

Function MultiplyNums (d,a)
  MultiplyNums = d * a
End Function

Sub DisplayMessage (msg)
  MsgBox msg
End Sub

Function GetName
  GetName = InputBox("Enter Your Name : ")
End Function

VBEND

Let>a=5
VBEval>MultiplyNums(%a%,2),answer
MessageModal>answer

VBRun>DisplayMessage,Hello World

VBEval>GetName,name
MessageModal>Your Name is : %name%
```

4.24.4 VBRun

VBRun>Subroutine[,Parm1,Parm2,...]

Not supported in Macro Scheduler Lite.

Executes the specified VBScript subroutine in the proceeding VBScript code block. Parameters can be passed to the subroutine. Macro Scheduler allows up to 25 parameters to be passed. The number of parameters passed must equal the number expected by the subroutine, otherwise a run time error will occur.

Macro Scheduler variables may be used in any part of the command.

© 2015 MJT Net Ltd

It is possible to set the VBScript code timeout using the VBS_TIMEOUT variable. By default VBScript code will never timeout. To set a timeout, set VBS_TIMEOUT to a value in milliseconds.

Abbreviation : VBR
See also: VBEval 226
Links to VBScript Documentation: http://www.mjtnet.com/resources.htm

Example

```
VBSTART

Function MultiplyNums (d,a)
  MultiplyNums = d * a
End Function

Sub DisplayMessage (msg)
  MsgBox msg
End Sub

Function GetName
  GetName = InputBox("Enter Your Name : ")
End Function

VBEND

Let>a=5
VBEval>MultiplyNums(%a%,2),answer
MessageModal>answer

VBRun>DisplayMessage,Hello World

VBEval>GetName,name
MessageModal>Your Name is : %name%
```

4.25 WebRecorder Functions

4.25.1 IEClickTag

IEClickTag>IE_Reference,Frame,Form,Tagname,Attribute,Value,result

Simulates clicking an html element.

IE_Reference: The identifier of the instance to use. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: The name of the frame containing the element

Form: If a form object, the name of the form containing the element. Can be empty.

Tagname: The element's tag name (e.g. 'A', 'IMG', 'INPUT')

Attribute takes any tag attribute, or one of the following values:

© 2015 MJT Net Ltd

'TEXT' : identifies the element by the element's innertext property. If element's innertext matches value the element is clicked.
'HREF' : identifies the element by the href property. If element's href matches value the element is clicked.
'NAME' : identifies the element by its name property. If element's name matches value the element is clicked. In this case the element must also belong to the specified form.
'SRC' : identifies the element by the src property. If element's src property matches value the element is clicked.
'COORDS': for Area objects. If area's coords proprty matches value the element is clicked.
'RADIO' : For Radio objects. If element belongs to specified form and value matches "element_name:element_value" the element is clicked.
'CHECKBOX': For checkboxes. Works as with RADIO.
'TITLE' : identifies an 'A' element by its title property. If element's title matches value the element is clicked.
'ID' : identifies an element by its ID property. If elements ID matches the value specified the element is clicked.
'INDEX' : identifies an element by its zero based numeric index. This can be used if no other attribute can be used to identify the element uniquely. E.g. an INDEX of 0 for an A element will click the first link on the page, INDEX of 1 the next, and so on.
'CLASSNAME': identifies an element by its CLASSNAME property. If element's CLASSNAME matches the value specified the element is clicked.
Alternatively any other tag attribute can be used.

See also: WebRecorder Functions 229

Example

```
//Click A tag with text "Log In" in IE[0] instance
IEClickTag>IE[0],,,A,TEXT,Log In,result
```

4.25.2 IEContainsText

IEContainsText>IE_Reference,Frame,Text,Result

Determines whether or not text is contained in the body of the page. Returns 1 if text is found, 0 if not.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Specify a frame name to search within a specific frame. Leave frame blank if there are no frames.

See also: WebRecorder Functions 229

Example

```
IEContainsText>IE[0],,Hello World,exists
```

4.25.3 IECreate

IECreate>New_IE_Reference_Variable

Creates a new IE instance, returning the instance handle in the given reference variable. Use this

© 2015 MJT Net Ltd

variable with subsequent IE functions to target the correct browser window.

See also: WebRecorder Functions 229

Example

```
//Return new IE instance handle in IE0 variable
IECreate>IE[0]
```

4.25.4 IEExtractTable

IEExtractTable>IE_Reference,Frame,Index,Filename,RemoveCRLFs,Result

Extracts a table to a CSV file. Specify the number of the table on the page in index.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Specify the frame name, or leave frame blank if no frame (top level document).

To remove carriage returns and line feeds from the output data set RemoveCRLFs to 1. Otherwise set RemoveCRLFs to 0.

Returns 0 if successful, 1 if an error occured.

See also: WebRecorder Functions 229

Example

```
IEExtractTable>IE[0],,21,c:\my documents\table.csv,1,r
```

4.25.5 IEExtractTableByName

IEExtractTableByName>IE_Reference,Frame,Name_Or_ID,Filename,RemoveCRLFs,Result

Extracts a table to a CSV file. Specify the name or ID of the table on the page in name_or_id.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Specify the frame name, or leave frame blank if no frame (top level document).

To remove carriage returns and line feeds from the output data set RemoveCRLFs to 1. Otherwise set RemoveCRLFs to 0.
Returns 0 if successful, 1 if an error occured.

See also: WebRecorder Functions 229

Example

```
IEExtractTableByName>IE0,,customers,c:\my documents\table.csv,1,r
```

© 2015 MJT Net Ltd

4.25.6 IEExtractTag

IEExtractTag>IE_Reference,Frame,Tagname,Index,All,Result

Extracts text from the specified tag.

IE_Reference: The web browser instance to extract from. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame to extract from

Tagname: the html name of the tag to extract from (e.g. 'TD', 'BODY', 'A', 'P', 'TABLE', etc)

Index: the index of the tag with tagname to extract from.

All: if set to 1 extract all html, otherwise just innertext, or value for input fields.

See also: WebRecorder Functions 229

Example

```
IEExtractTag>IE[0],,TD,10,0,strTDContents
//Extracted text is in variable: strTDContents
```

4.25.7 IEExtractTagByAttrib

IEExtractTagByAttrib>IE_Reference,Frame,Tagname,Attrib,Value,All,Result

Extracts text from the specified tag.

IE_Reference: The web browser instance to extract from. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame to extract from

Tagname: the html name of the tag to extract from (e.g. 'TD', 'BODY', 'A', 'P', 'TABLE', etc)

Attrib: any valid tag attribute

Value: the value of the specified tag attribute

All: if set to 1 extract all html, otherwise just innertext, or value for input fields.

See also: WebRecorder Functions 229

Example

```
IEExtractTagByAttrib>%IE[0]%,{""},{"INPUT"},{"NAME"},{"q"},1,buffer
MessageModal>buffer
```

© 2015 MJT Net Ltd

4.25.8 IEExtractTagByName

IEExtractTagByName>IE_Reference,Frame,Tagname,Name_Or_ID,All,Result

Extracts text from the specified tag.

IE_Reference: The web browser instance to extract from. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame to extract from

Tagname: the html name of the tag to extract from (e.g. 'TD', 'BODY', 'A', 'P', 'TABLE', etc)

Name_or_id: the name or id of the tag with tagname to extract from.

All: if set to 1 extract all html, otherwise just innertext, or value for input fields.

Result: the variable to return the text in

If the element is a checkbox and is checked ":CHECKED" will be appended to the result which will also contain the checkbox's value. i.e. result will contain "value:CHECKED".

See also: WebRecorder Functions 229

Example

```
IEExtractTagByName>IE[0],,DIV,sidebar,0,divside
MessageModal>Data: %divside%
```

4.25.9 IEFormFill

IEFormFill>IE_Reference,Frame,Form,Fieldname,Value,Event,Result

Fills a form field.

IE_Reference: The identifier of the web browser instance to use. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame containing the form field object

Form: the name of the form (can be blank if the form has no name). Can also be set to "ACTION:action_script" where action_script is the action of the form (e.g. search.htm).

Fieldname: the name of the field to fill. If the field has no name but has an ID attribute, the ID attributed can be specified instead. Alternatively you can specify the className of an element by issuing CLASSNAME:class where class is the className of the field.

Value: the value to set the field value to.

Event: optional parameter that can take one of the following values:
submit: forces the form to submit.
onchange: triggers the field's onchange handler if it has one.

If the field is a SELECT object it is possible to select an item based on it's index, rather than actual value, by specifying fieldvalue as:

© 2015 MJT Net Ltd

#INDEX#:n where n is the zero based index of the item. To select all items in a multi-select SELECT object use #INDEX#:ALL

Returns the number of fields that matched the specified values if successful, 0 if unsuccessful.

See also: WebRecorder Functions 229

Example

```
IEFormFill>IE[0],,main_form,firstname,Fred,0,r
```

4.25.10 IEFormSubmit

IEFormSubmit>IE_Reference,Frame,Form,Result

Submits a form. Usually this is done by clicking a Submit button or a link and therefore this function will not always be necessary as you'd probably use an IE_ClickTag instead. However, sometimes forms may be submitted in a different way, such as when the Enter key is pressed in the last field of the form. IE_FormSubmit can be used. Another way to do this is just to set the event parameter of IEFormFill to submit.

IE_Reference: The identifier of the web browser instance to use. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame containing the form field object

Form: the name of the form (can be blank if the form has no name). Can also be set to "ACTION:action_script" where action_script is the action of the form (e.g. search.htm).

See also: WebRecorder Functions 229

Example

```
IEFormFill>%IE[0]%,{""},{"ACTION:search.htm"},{"q"},{"fred"},0,ie_res
IEFormSubmit>%IE[0]%,{""},{"ACTION:search.htm"},ie_res
```

4.25.11 IEGetFromURL

IEGetFromURL>URL,IE_Reference

Returns a reference to an existing IE web browser instance. The instance is identified by the URL. URL can be full or partial. The first IE web browser document found whose URL contains or matches the URL specified will be used. The returned IE_Instance can be used with other WebRecorder functions.

See also: WebRecorder Functions 229

Example

```
IEGetFromURL>www.somewhere.com,IE_Inst
IEClickTag>IE_Inst,,,A,TEXT,Log In,result
```

© 2015 MJT Net Ltd

4.25.12 IEGetHWND

IEGetHWND>IE_Reference,Result

Returns the window handle of the specified IE instance.

IE_Reference must be an identifier value returned by IECreate 230 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
IEGetHwnd>IE[0],ie_handle
```

4.25.13 IEGetTagPos

IEGetTagPost>IE_Reference,Frame,Tagname,Attrib,Value,Xpos,Ypos

Retrieves the screen X,Y position of the matching tag.

IE_Reference: The identifier of the instance to use. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: The name of the frame containing the element

Tagname: The element's tag name (e.g. 'A', 'IMG', 'INPUT')

Attrib: the identifying attribute of the element e.g. innerText or ID. (TEXT can be used for innerText). Alternatively this can be INDEX to identify the element by 0-based numeric index where no unique attributes exist.

Value: the value of the given attribute.

Note that the position returned will be the top left position of the element. You may need to apply a small offset for a mouse over to be triggered.

See also: WebRecorder Functions 229

Example

```
IEGetTagPosX>%IE[2]%,{""},{"A"},{"TEXT"},{"Home"},tagX,tagY
MouseMove>tagX,tagY
```

4.25.14 IEGetURL

IEGetURL>IE_Reference,Result

Returns the URL of the specified IE instance. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

© 2015 MJT Net Ltd

```
IEGetURL>IE[0],ie_handle
```

4.25.15 IEGoBack

IEGoBack>IE_Reference,Result

Navigates the given IE web browser instance back one page. Same as clicking the back button. Returns zero if successful, one if an error occurs.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
//Go back one page in IE[0] instance
IEGoBack>IE[0],result
```

4.25.16 IEGoForward

IEGoForward>IE_Reference,Result

Navigates the given IE web browser instance forward one page. Same as clicking the forward button. Returns zero if successful, one if an error occurs.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
//Go forward one page in IE[0] instance
IEGoForward>IE[0],result
```

4.25.17 IENavigate

IENavigate>IE_Reference,URL,Result

Navigates the specified browser instance to URL. Returns 0 if successful, 1 if not.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
//Navigate IE0 instance
IENavigate>IE[0],http://www.mjtnet.com/,ie_res
```

© 2015 MJT Net Ltd

4.25.18 IEOnDownload

IEOnDownload>Enabled,Path,Result

Enables or disables Macro Scheduler's IE download manager.

Enabled: 1 to enable, 0 to disable

Path: the folder where files should be downloaded to

Internet Explorer's default file download manager is difficult to automate as it requires user interaction. This function can be used to tell IE to automatically download files to the specified path without requiring user confirmation. This makes it much easier to script file downloads.

WebRecorder will automatically start all scripts with IE_OnDownload enabling Macro Scheduler's download manager.

When WebRecorder detects a file download it will also output the filename and size to script variables and insert an IEWaitFileDownload 239 call so that when played back the script will wait until the file download is complete before continuing.

Note that sometimes the filename that is returned may need renaming for the file to function or may not be what you want. This is particularly the case where a download script redirects to the actual download. You can simply use a RenameFile 128 command after the IEWaitFileDownload 239 call to rename the file to whatever you want it to be.

See also: WebRecorder Functions 229

Example

```
IEOnDownload>1,C:\Documents and Settings\Marcus\My Documents,ie_res
IECreate>IE[0]
IENavigate>%IE[0]%,www.somewhere.com/somefile.exe,ie_res
Let>downloaded_filename=C:\Documents and Settings\My Documents\somefile.exe
Let>downloaded_filesize=10582752
IEWaitFileDownload>somefile.exe,ie_res
```

4.25.19 IEQuit

IEQuit>IE_Reference,result

Closes the specified IE window.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
//Quit IE[0] instance
IEQuit>IE[0],ie_res
```

© 2015 MJT Net Ltd

4.25.20 IERefresh

IERefresh>IE_Reference,Result

Refreshes the page - does the same as clicking the refresh button on the web browser specified by it's identifier. Returns 0 if successful, 1 if an error occurred.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

```
//Go forward one page in IE[0] instance
IERefresh>IE[0],result
```

4.25.21 IESetTimeout

IESetTimeout>Timeout,Result

Sets the timeout in seconds to use for IEClickTag 229, IEFormFill 233 and IEWaitDocumentComplete 239

To help ensure IEAuto does not try to click or fill tags before they exist IE_ClickTag 229 and IE_FormFill 233 will wait until they exist, or timeout within the time specified by IE_SetTimeout. IE_WaitDocumentComplete 239 also uses this timeout to prevent hangs should a page fail to load.

See also: WebRecorder Functions 229

Example

```
//Set timeout for ClickTag and FormFill to 10 seconds, increase if pages load more slowly
IESetTimeout>10,ie_res
```

4.25.22 IEShowIE

IEShowIE>IE_Reference,Show

ShowIE allows you to show or hide an IE instance. Set Visible to 0 to hide the instance, and 1 to show it. IECreate 230 always creates IE in visible mode so use IEShowIE immediately after IECreate 230 to hide it.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

See also: WebRecorder Functions 229

Example

//Hide IE[0] instance

```
IEShowIE>IE[0],0
```

© 2015 MJT Net Ltd

4.25.23 IEWait

IEWait>IE_Reference/URL

Waits for the specified browser instance to no longer be busy by monitoring IE's busy property. (Can take a URL instead of IE instance).

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241 or a URL.

Note that waiting for IE to not be busy is not necessarily the same as waiting for the overall document to be complete. For waiting for document completeness we recommend using IEWaitDocumentComplete 239.

See also: WebRecorder Functions 229

Example

```
//Wait for IE[0] instance
IEWait>IE[0]
```

4.25.24 IEWaitDocumentComplete

IEWaitDocumentComplete>IE_Reference,Result

Waits for the document in the given web browser instance to have completely finished loading.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

This function will timeout if the value set by IESetTimeout 238 elapses before the document has loaded. It will return 0 if it times out, 1 if successful.

This is output by WebRecorder automatically after every document load.

See also: WebRecorder Functions 229

Example

```
IENavigate>%IE[0]%,www.mjtnet.com,ie_res
IEWaitDocumentComplete>%IE[0]%,ie_res
```

4.25.25 IEWaitFileDownload

IEWaitFileDownload>Filename,Result

Waits for a file download operation for the given file to complete. This code is automatically output by WebRecorder when a file download takes place.

See also: WebRecorder Functions 229

Example

```
IEOnDownload>1,C:\Documents and Settings\Marcus\My Documents,ie_res
```

© 2015 MJT Net Ltd

```
IECreate>0,IE[0]
IENavigate>%IE[0]%,www.somewhere.com/somefile.exe,ie_res
Let>downloaded_filename=C:\Documents and Settings\My Documents\somefile.exe
Let>downloaded_filesize=10582752
IEWaitFileDownload>somefile.exe,ie_res
```

4.25.26 IEWaitForText

IEWaitForText>IE_Reference,Frame,Text,All,Timeout,Result

Waits for the specified text to exist in the body of the page before continuing.

IE_Reference: The web browser instance to extract from. IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Frame: the name of the frame to extract from

Text: the text to wait for

All: if set to 1 look in all html otherwise just text

Timeout: number of seconds to wait

Returns 1 if the text was found within the given timeout seconds. If it times out it will return 0.

See also: WebRecorder Functions 229

Example

```
IEWaitForText>IE[0],,{"Log in"},0,20,result
```

4.25.27 IEWaitTimeout

IEWaitTimeout>IE_Reference,Timeout_Seconds,Post_Seconds,Result

Waits for the specified browser instance to no longer be busy by monitoring IE's busy property. Will timeout if timeout_seconds seconds elapses before IE returns not busy. Returns 1 if the command timed out before the browser ceased being busy.

IE_Reference must be an identifier value returned by IECreate 230, IEGetFromURL 234 or IEWaitNew 241.

Waits post_seconds afterwards.

Note that waiting for IE to not be busy is not necessarily the same as waiting for the overall document to be complete. For waiting for document completeness we recommend using IEWaitDocumentComplete 239

See also: WebRecorder Functions 229

Example

```
//Wait for IE[0] instance with 30 second timeout, and 1 second post delay
IEWaitTimeout>IE[0],30,1,res
```

© 2015 MJT Net Ltd

4.25.28 IEWaitNew

IEWaitNew>New_IE_Reference

Waits for a new IE WebBrowser object to be created. This is needed where a call to one of the other functions, such as IEClickTag 229 or IEFormFill 233 results in a new Internet Explorer window being created. E.g. the response to a ClickTag could be a new popup. Call IEWaitNew to wait for this new instance to be created. Returns an identifier to the new instance. Reserved is undefined - set to zero.

See also: WebRecorder Functions 229

Example

```
IEWaitNew>IE[1]
```

4.26 Window Functions

4.26.1 CloseWindow

CloseWindow>window_title

Attempts to close the specified window. The window_title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

Please note: this command simply sends the internal Windows WM_CLOSE message to the target window. This is the internal instruction to tell the window to close. However, applications all interpret this instruction differently and there is no guarantee that it will cause the window to close. If CloseWindow fails to close the window we recommend simulating user input to close it instead (e.g. by sending ALT-F4 or equivalent).

If the specified window is the main window of an application, then that application will begin to close down. Any processing that the application does on exit will be carried out as usual.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to close the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

© 2015 MJT Net Ltd

Abbreviation : Clo
See also: GetWindowPos 245, GetActiveWindow 242

Example

```
CloseWindow>notepad*
```

4.26.2 FindWindowWithText

FindWindowWithText>text_to_find,setfocus_flag,result_variable

This function attempts to locate a window containing somewhere within it the text specified in text_to_find. setfocus_flag can be set to 1 or 0. If set to 1, the function will activate the window it finds containing the specified text. If a window is found, result_variable will contain the found window's title text. If no window is found it will contain "NOT FOUND".

If the WIN_USEHANDLE variable is set to 1 result_variable will contain the window's handle.

NB. Not all text can be detected successfully. Some applications, such as Word Processors, in order to provide their more complex functionality have to display text in the form of graphics, so do not manifest textual data to other applications. FindWindowWithText uses the same routines as the View System Windows 36 tool. So to see what text can be detected at any particular time use the View System Windows 36 tool.

This function is useful when an application has two windows with the same name, and allows the correct one to be located and focused.

Abbreviation : Fin

Example

```
FindWindowWithText>Continue,1,windowname
```

4.26.3 GetActiveWindow

GetActiveWindow>window_title,X,Y[,Width,Height]

Retrieves information about the current active window. The window title, and top left coordinates of the active window are stored in window_title, X and Y.

If the WIN_USEHANDLE variable is set to 1 window_title will contain the window's handle.

The optional parameters Width and Height will return the width and height of the window respectively.

If GAW_TYPE is set to 1 (default is 0) this command will retrieve information about the active ***child*** window of the active foreground window. If there are no child windows window_title will be set to an empty string. For default behaviour set GAW_TYPE back to 0.

Abbreviation : GAW
See also: GetWindowPos 245, CloseWindow 241, GetWindowHandle 243

© 2015 MJT Net Ltd

4.26.4 GetWindowChildList

GetWindowChildList>parent_handle,child_list

Returns a list of child objects/windows belonging to the specified parent handle.

By default child_list returns a list of window titles/captions. If WIN_USEHANDLE it set to 1 child_list will be a list of handles.

4.26.5 GetWindowHandle

GetWindowHandle>window_title,handle

Returns the window handle of the window specified by it's window title.

Window handles are assigned when a window is created by the system so they change from session to session. Using window handles is more reliable than using window titles which are not unique and may require substring matching. You can use GetWindowHandle to retrieve the window handle and then refer to it subsequently in other window commands when you need to access that window by setting WIN_USEHANDLE to 1. See also GetActiveWindow 242 which can return the handle of the currently active window.

The window_title may contain the * symbol at the end to indicate a substring match.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation: GWH
See also: GetActiveWindow 242

Example

```
GetWindowHandle>Notepad*,notepad_hwnd
Let>WIN_USEHANDLE=1
SetFocus>notepad_hwnd
```

4.26.6 GetWindowList

GetWindowList>window_list

Returns a list of open top-level window titles. The list is delimited by carriage-return-line-feed pairs (CRLF).

To return window handles instead of window titles first set the WIN_USEHANDLE variable to 1.

© 2015 MJT Net Ltd

Abbreviation: GWL

Examples:

```
//Loop through a list of windows
GetWindowList>winlist
Separate>winlist,CRLF,windows
Let>k=1
Repeat>k
  Let>this=windows_%k%
  MessageModal>this
  Let>k=k+1
Until>k>windows_count

//Populate a dialog combo box with a window list
GetWindowList>Dialog1.msComboBox1.Items.Text
ResetDialogAction>Dialog1
```

4.26.7 GetWindowNames

GetWindowNames>handle,title,class

Given a window handle GetWindowNames will return the window's title caption and class name in the title and class variables.

Abbreviation: GWN

Example

```
Let>WIN_USEHANDLE=1
GetWindowList>allWindows
Separate>allWindows,CRLF,wins
If>wins_count>0
Let>k=0
Repeat>k
  Let>k=k+1
  Let>this_window=wins_%k%
  GetWindowNames>this_window,strTitle,strClass
  // ....
Until>k=wins_count
```

4.26.8 GetWindowParent

GetWindowParent>handle,parent_type,result

Returns the parent handle of the object with the given handle. parent_type can be one of the following:

1: Retrieve the parent window
2: Retrieve the root parent window
3: Retrieve the root owner window

© 2015 MJT Net Ltd

4.26.9 GetWindowPos

GetWindowPos>window_title,X,Y

Locates the window specified in window_title and retrieves its upper left screen coordinates. X and Y are variables in which to store the coordinates. window_title may be a variable or literal.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

The window_title may contain the * symbol at the end to indicate a substring match.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : GWP
See also: GetActiveWindow 242, CloseWindow 241, GetWindowSize 246

Example

```
GetWindowPos>My Computer,XWindowPos,YWindowPos
```

The following example achieves the same result as the MouseMoveRel command, moving to the point 10,10 relative to Notepad :

```
GetWindowPos>notepad*,npX,npY
Add>npX,10
Add>npY,10
MouseMove>npX,npY
```

4.26.10 GetWindowProcess

GetWindowProcess>window_title,process_id,process_name

Retrieves the process ID and process name (filename) of the process that the specified window belongs to.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

The window_title may contain the * symbol at the end to indicate a substring match.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing

© 2015 MJT Net Ltd

asterisk will force Macro Scheduler to only look for an exact match.

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation: GPW

Example

```
GPW>Notepad*,pid,name
MessageModal>%pid% %name%
```

4.26.11 GetWindowSize

GetWindowSize>window_title,Width,Height

Locates the window specified in window_title and retrieves its width and height dimensions.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

The window_title may contain the * symbol at the end to indicate a substring match.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : GWS
See also: GetActiveWindow 242, GetWindowPos 245

Example

```
GetWindowSize>My Computer,nWidth,nHeight
MessageModal>Dimensions: %nWidth%,%nHeight%
```

4.26.12 GetWindowText

GetWindowText>window_title,text

GetWindowText retrieves all the detectable text contained within the specified window. Specify the window using the exact window title. The text is retrieved as a list with each object's text on a new line.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

GetWindowText retrieves the published text property of the specified window/object. Not all text that you see on the screen is retrievable in this way. Some text is painted via lower level routines and some text is graphical. Some objects, such as labels, are not windowed controls, and therefore text

© 2015 MJT Net Ltd

associated with them cannot be retrieved with GetWindowText. Try the new GetTextAtPoint, GetTextInRect and GetWindowTextEx commands which use lower level hooks to trap more text.

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : GWT
See also: GetWindowTextEx 225

Example

```
GetWindowText>Document - WordPad,WordPadText
```

4.26.13 IfWindowOpen

IfWindowOpen>window_title[,label_name[,false_label_name]]
statements
[[Else
*else statements***]**
Endif]

Checks to see if the specified window is open. If so the first statements are executed. Otherwise the else statements are executed. Optionally, instead of Else and Endif a label or subroutine name can be specified.

When label names are specified the command causes the script to continue from the specified label without running any other lines of code in between. If a second false label is specified, execution will jump to that label if the expression resolves false. Subroutine names can also be specified. When specifying a subroutine name execution will jump to that subroutine and return to the line after the If statement when the subroutine has completed.

The window_title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command effects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : IfW
See also: Label 202, Goto 200, IfFileExists 124, IfFileChanged 124, If 146, Subroutines 197

Example

```
IfWindowOpen>Notepad - [Untitled]
```

© 2015 MJT Net Ltd

```
  ...
Endif
```

or, with a wildcard :

```
IfWindowOpen>notepad*
  ...
Endif
```

4.26.14 IfNotWindowOpen

IfNotWindowOpen>window_title[,label_name[,false_label_name]]
statements
[[Else
else statements]
Endif]

The opposite of IfWindowOpen 247 but with the boolean logic reversed. See IfWindowOpen 247 for usage.

4.26.15 MoveWindow

MoveWindow>window_title,new_X,new_Y

Moves the window with the specified window title to the new coordinates. new_X,new_Y denotes the new upper left corner position. The window_title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to move the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : MVW
See also: ResizeWindow 249

Example

```
MoveWindow>notepad*,5,5
```

© 2015 MJT Net Ltd

4.26.16 ResizeWindow

ResizeWindow>window_title,new_width,new_height

Resizes the window with the specified window with the new width and height dimensions. The window_title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to resize the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : RSW
See also: MoveWindow 248

Example

```
ResizeWindow>notepad*,500,300
```

4.26.17 SetFocus

SetFocus>window_title

Sets focus to the specified window. The window_title may contain the * symbol **at the end** to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to setfocus to the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and sets focus to the first one it finds whose title **contains** the entered text (using a case insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

To summarise:

SetFocus>window_title
Searches for a window title matching window_title fully and case-sensitively.

© 2015 MJT Net Ltd

SetFocus>window_title*
First performs an exact case-sensitive match as above and if nothing found does a case-insensitive substring search where window_title can be anywhere within the matching window title.

For more control regular expressions can be used if WIN_REGEX is first set to 1.

To set focus to child windows, such as MDI children, first use SetFocus to focus the application, and then issue a second SetFocus for the child window. In most cases it is necessary to add the * symbol for child windows.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

Abbreviation : Set

Example

```
SetFocus>notepad*
```

Child Window Example:

```
SetFocus>Opera*
SetFocus>google*
```

RegEx Example:

```
//find first matching window that starts with 1 or more printable characters and ends with "Not
Let>WIN_REGEX=1
SetFocus>.+Notepad$
```

4.26.18 ShutDownWindows

ShutDownWindows>shutdown_type

Use this command to shutdown or reboot the machine. Set shutdown_type appropriatly as follows:

0: Shutdown
1: Reboot
2: Logoff
3: Forced Shutdown
4: Forced Reboot
5: Forced Logoff

The last three options should be used with care as they will force running processes to terminate and this can cause applications to lose data as they are not being allowed to close down gracefully.

Abbreviation : SDW

Example

```
//Reboot Server
ShutDownWindows>1
```

© 2015 MJT Net Ltd

4.26.19 WaitWindowChanged

WaitWindowChanged>timeout

This command causes Macro Scheduler to wait until the foreground window changes. I.e. the foreground window's title (caption) changes, or a different window becomes the foreground window. If it doesn't change within the number of seconds specified in Timout, the command stops waiting and the variable WWC_RESULT is set to FALSE. WWC_RESULT is TRUE if the command terminated because the foreground window changed within the specified time. if Timeout is set to 0, the command will wait indefinitely.

Abbreviation : WWX
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitWindowFocused 252

4.26.20 WaitWindowClosed

WaitWindowClosed>window_title

Waits for a specified window to close. Execution of the script will not continue until the window with the specified title text is no longer present or a specified timeout value is exceeded. The window title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

The system variable WW_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WW_TIMEOUT is used, WW_RESULT will indicate whether or not the command ended successfully. If it timed out WW_RESULT will be set to FALSE. If the window it was waiting for closed within the timeout setting, the WW_RESULT value will be set to TRUE.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

By definition this function continually enumerates and loops through the list of windows open on the system. This is an intensive process and you may therefore see CPU usage increase during the wait. For most people this is not a problem. However, if you wish to reduce CPU usage you can set WIN_SLEEP to 1 at the expense of processing time (in theory setting WIN_SLEEP to 1 will slow the function down while reducing CPU usage).

© 2015 MJT Net Ltd

Abbreviation : WWC
See also: Wait 209, WaitWindowOpen 253, WaitWindowChanged 251, WaitWindowFocused 252.

Examples

```
WaitWindowClosed>Progress

//With a timeout check
Let>WW_TIMEOUT=10
WaitWindowClosed>Progress
If>WW_RESULT=FALSE
  MessageModal>Timedout
Endif
```

4.26.21 WaitWindowFocused

WaitWindowFocused>window_title

Waits for a specified window to open/appear and be focused (the foreground window). Execution of the script will not continue until a window with the specified title text appears and is focused or a specified timeout value is exceeded. The window title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

The system variable WW_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WW_TIMEOUT is used, WW_RESULT will indicate whether or not the command ended successfully. If it timed out WW_RESULT will be set to FALSE. If the window it was waiting for appeared within the timeout setting, the WW_RESULT value will be set to TRUE.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

By definition this function continually enumerates and loops through the list of windows open on the system. This is an intensive process and you may therefore see CPU usage increase during the wait. For most people this is not a problem. However, if you wish to reduce CPU usage you can set WIN_SLEEP to 1 at the expense of processing time (in theory setting WIN_SLEEP to 1 will slow the function down while reducing CPU usage).

Abbreviation : WWF
See also: Wait 209, WaitWindowOpen 253, WaitWindowClosed 251, WaitWindowChanged 251

© 2015 MJT Net Ltd

Examples

```
Run Program>c:\program files\msoffice\winword.exe
WaitWindowFocused>microsoft word*

//With timeout checking
Let>WW_TIMEOUT=30
WaitWindowFocused>microsoft word*
If>WW_RESULT=FALSE
  MessageModal>Error starting Word!
Endif
```

4.26.22 WaitWindowOpen

WaitWindowOpen>window_title

Waits for a specified window to open/appear. Execution of the script will not continue until a window with the specified title text appears or a specified timeout value is exceeded. The window title may contain the * symbol at the end to indicate a substring match.

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will first attempt to find the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

The system variable WW_TIMEOUT can be used to set the number of seconds after which this command should timeout. If set to zero (the default) the timeout will not occur and the command will continue indefinitely. If WW_TIMEOUT is used, WW_RESULT will indicate whether or not the command ended successfully. If it timed out WW_RESULT will be set to FALSE. If the window it was waiting for appeared within the timeout setting, the WW_RESULT value will be set to TRUE.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

By definition this function continually enumerates and loops through the list of windows open on the system. This is an intensive process and you may therefore see CPU usage increase during the wait. For most people this is not a problem. However, if you wish to reduce CPU usage you can set WIN_SLEEP to 1 at the expense of processing time (in theory setting WIN_SLEEP to 1 will slow the function down while reducing CPU usage).

Abbreviation : WWO
See also: Wait 209, WaitWindowClosed 251, WaitWindowChanged 251, WaitWindowFocused 252

Examples

© 2015 MJT Net Ltd

```
Run Program>c:\program files\msoffice\winword.exe
WaitWindowOpen>microsoft word*

//With timeout checking
Let>WW_TIMEOUT=30
WaitWindowOpen>microsoft word*
If>WW_RESULT=FALSE
  MessageModal>Error starting Word!
Endif
```

4.26.23 WindowAction

WindowAction>Action,window_title

Use this function to restore, minimize, maximize or close a window.

Action can be any one of the following:

0: Restore the window
1: Maximize the window
2: Minimize the window
3: Close the window

If the WIN_USEHANDLE variable is set to 1 window_title must be a window handle.

Specify the window name in window_title. The window_title may contain the * symbol at the end to indicate a substring match.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to select the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and selects the first one it finds whose title **contains** the entered text (using a case-insensitive search). This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

It is possible to limit the type of windows this command affects using the WF_TYPE variable:
Let>WF_TYPE=0 - No Child Windows
Let>WF_TYPE=1 - ALL Windows (Default)
Let>WF_TYPE=2 - Visible Windows Only
Let>WF_TYPE=3 - Child Windows Only

A Regular Expression can be used in window_title if WIN_REGEX is set to 1.

Abbreviation : WIN
See also: MoveWindow 248, ResizeWindow 249, CloseWindow 241

Example

```
WindowAction>2,notepad*
```

© 2015 MJT Net Ltd

4.27 Window Objects

4.27.1 FindObject

FindObject>parent_handle,ClassName,CaptionText,Instance,return_handle,X1,Y1,X2,Y2,return_text

Locates a control given its parent's handle, its class name and either its caption text or instance. Returns the control's handle, its rectangular bounding screen coordinates and any published caption text. If an Index is specified the caption text value is ignored. Where only one object with the given class name exists on the parent window/object instance would be 1. Where 2 exists, 1 would return the first item and 2 the 2nd. The order matches the order shown in the View System Windows 36 tool.

FindObject can be used in conjunction with SendObjectText and SendObjectKeys to send keystrokes directly to a control. Use the "Send Keys To Object" wizard to locate the control and generate the code automatically.

See also: SendObjectText, SendObjectKeys, GetControlText 256, SetControlText 261, SetObjectText 261

Examples:

```
//Get the text from Notepad's editor:
GetWindowHandle>Untitled - Notepad,hWnd
FindObject>hWnd,Edit,,1,hWnd,X1,Y1,X2,Y2,result

//Locates the Windows start button, returning its position and caption text
FindObject>0,Shell_TrayWnd,,1,hWnd,X1,Y1,X2,Y2,result
FindObject>hWnd,Button,,1,hWnd,X1,Y1,X2,Y2,result

//Send keystrokes directly to Notepad's edit control (Notepad does not require focus)
GetWindowHandle>Untitled - Notepad,hWndParent
FindObject>hWndParent,Edit,,1,hWnd,X1,Y1,X2,Y2,result
ObjectSendKeys>hWnd,{"Hello World"}
```

4.27.2 GetCheckBox

GetCheckBox>window_title,object_caption,result

GetCheckBox determines whether or not the given check box, or radio button, is checked. result is set to 1 if checked, 0 if not checked, or -1 if the command failed to locate the given check box. Specify the window title of the window containing the check box/radio button, and the object's caption. The caption of a check box or radio button is the text appearing next to it. The caption must be specified accurately with attention paid to case. Where a letter is underlined indicating a shortcut key, enter the '&' character before it.

window_title can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation : CBX

© 2015 MJT Net Ltd

See also: SetCheckBox 260

Example

```
GetCheckBox>Internet Explorer Prop*,Never dial a &connection,res
If>res=1
  MessageModal>The checkbox is checked!
Else
  MessageModal>The checkbox is not checked!
Endif
```

4.27.3 GetControlText

GetControlText>WindowTitle,ClassName,Instance,Result

Returns the text of the control specified by ClassName and Instance on the window specified by WindowTitle.

Result contains the text of the control if found. It the window is not found Result contains ##NOSUCHWINDOW##. If the class and instance was not found ##NOSUCHOBJECT## is returned.

If WIN_USEHANDLE is set to 1 WindowTitle must be a window handle.

To determine class names of objects on windows use View System Windows 36. A window may contain several objects of the same class name. Instance is used to determine which instance of the class to use.

GetControlText retrieves the published text property of the specified object. Not all text that you see on the screen is retrievable in this way. Some text is painted via lower level routines and some text is graphical. Some objects, such as labels, are not windowed controls, and therefore text associated with them cannot be retrieved with GetControlText. Try the new GetTextAtPoint, GetTextInRect and GetWindowTextEx commands which use lower level hooks to trap more text.

WintowTitle can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation: GCT
See also: SetControlText 261, GetTextAtPoint 222, GetTextInRect 223, GetWindowTextEx 225, WaitScreenText 225

Example:

```
GetControlText>Notepad*,Edit,1,npEdit
```

npEdit will now contain the contents of the notepad edit window.

4.27.4 GetFocusedObject

GetFocusedObject>handle

Returns the handle of the currently focused object.

© 2015 MJT Net Ltd

Abbreviation: GFO

4.27.5 GetListItem

GetListItem>WindowTitle,ClassName,Instance,Text,Column,Case,Partial,Result,Handle

Returns in Result the row index of the text specified in Text in the specified ListView (List Box) object.

WindowTitle: The title of the window containing the ListView object
ClassName: Usually SysListView32 but may differ. Use View System Windows to determine class name.
Instance: The zero based index of the instance to use (a window may contain several listview objects).
Text: The text to search for in the list box.
Column: The column number to search in. 0 for first column or if only one column.
Case: 1 to perform a case sensitive match, 0 for non-case sensitive.
Partial: 1=partial, 0=full. A partial match will match Text at the **start** of the item text.
Result: Gets set to the zero based index of the item or -1 if not found.
Handle: Gets set to the handle of the ListView object.

The zero based index is returned in Result. Handle will be set to the handle of the ListView object.

Some list boxes allow you to "drill down" to the item you want by typing the value of the item. To select an item in these kinds of boxes all you therefore need to do is send keystrokes to them using the Send 162 command. However, some list boxes do not "drill down" and this is where GetListItem is needed. You may know the item text you want to select but can not predict it's index. Use GetListItem to retrieve the index and then you know how many times to "Press Down".

WindowTitle can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation: GLI
See also: GetTreeNode 258

Example

This example uses GetListItem to retrieve the index of an item in the Display Properties backgrounds box.

© 2015 MJT Net Ltd

```
//Open Display Properties
Run>rundll32 shell32.dll,Control_RunDLL Desk.cpl
WaitWIndowOpen>Display Properties

//Select Desktop Tab
Press CTRL
Press TAB
Release CTRL
WaitReady>0

//Get listindex of "Azul" in Background box
GetListItem>Display Properties,SysListView32,0,Azul,0,0,0,nDx,lvHwnd

//Select it
Press Home
Press Down * nDx

WaitReady>0

//OK!
Press Enter
```

4.27.6 GetTreeNode

GetTreeNode>WindowTitle,ClassName,Instance,Text,Case,Partial,Result,Handle

Returns in Result the index of the visible tree view node whose caption matches Text.

WindowTitle: The title of the window containing the TreeView object
ClassName: Usually SysTreeView32 but may differ. Use View System Windows to determine class name.
Instance: The zero based index of the instance to use (a window may contain several treeview objects).
Text: The text to search for in the list box.
Case: 1 to perform a case sensitive match, 0 for non-case sensitive.
Partial: 1=partial, 0=full. A partial match will match Text at the **start** of the item text.
Result: Gets set to the zero based index of the item or -1 if not found.
Handle: Gets set to the handle of the TreeView object.

The zero based index of the node is returned in Result. Handle will be set to the handle of the ListView object. Only visible nodes are searched.

WindowTitle can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation: GTN
See also: GetListItem 257

Example

This example uses GetTreeNode to retrieve the index of an item in Explorer's Folders treeview.

© 2015 MJT Net Ltd

```
//Assumes Windows Explorer is open at "My Documents" with
//explorer bar showing Folders
GetTreeNode>My Documents,SysTreeView32,0,My Computer,1,1,result,handle

Let>WIN_USEHANDLE=1
SetFocus>handle
Let>WIN_USEHANDLE=0

Press Home
Press Down * result
Press Right
```

4.27.7 PushButton

PushButton>window_title,button_caption

Attempts to 'click' the specified button of the specified window.

window_title can contain an asterisk (*) as with all other window functions. For buttons that have a hot key associated with them, and represented on the button by an underscored letter, pass a & character before that letter. e.g.: for a button called 'Close', send &Close.

This command works by attempting to send the BM_CLICK message to the button when it finds a button in the specified window with the given caption.

This will only work with objects of the standard 'Button' class. It may not work for all buttons on all windows. For instance, it doesn't work for buttons on html documents in Netscape Navigator 4.05 or Internet Explorer 3.02, as they are not real buttons. For these use the MouseOver 187 command.

If the last character of the window title specified is an asterisk (*), Macro Scheduler will attempt to locate the first window whose title matches the text entered exactly. If it cannot make an exact match it then looks at all windows and stops at the first one it finds whose title contains the entered text. This solves the problem with applications such as Word or Netscape which change their titles depending on the document loaded. It is best to try to provide an exact (including case) window title to ensure the correct window is found, as many applications have multiple invisible windows with similar names. Specifying text without a trailing asterisk will force Macro Scheduler to only look for an exact match.

window_title can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation : PUS
See also: MouseOver 187
Example

```
Run Program>rundll32.exe shell32.dll,Control_RunDLL TimeDate.cpl
WaitWindowOpen>Date/Time Properties
...
PushButton>Date/Time*,OK
```

© 2015 MJT Net Ltd

4.27.8 SelectMenu

SelectMenu>WindowTitle,MenuIndex[,SubMenuIndex[,SubMenuIndex[,...]]]

Selects the menu item specified by MenuIndex and optional SubMenuIndex parameters of the window specified by WindowTitle.

MenuIndex and SubMenuIndex are integer numeric values. For MenuIndex 0 is the first menu item on the left of the menu bar, 1 the next and so on. For SubMenuIndex, 0 is the first item in the submenu, 1 the next etc. The index includes separating lines, so remember to count them.

For example, to select the Save submenu item in Notepad, Save is the 3rd Submenu item of the File menu. The File menu is the 1st menu item in the menu bar. So MenuIndex is 0 and SubMenuIndex is 2. There are no further submenus. So the command would look like this:

```
SelectMenu>Notepad*,0,2
```

WindowTitle can contain the asterisk wildcard to indicate that it is a substring search (See SetFocus for more explanation). If the WIN_USEHANDLE variable is set to 1 WindowTitle must be a window handle rather than window title.

Abbreviation: MNU

Note: SelectMenu only works with standard Windows menus. Many menus are now in fact custom built menus that are other controls made to look like menus. SelectMenu will not therefore work with these objects.

WindowTitle can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

4.27.9 SetCheckBox

SetCheckBox>window_title,object_caption,TRUE|FALSE

SetCheckBox is used to check or uncheck a given checkbox or radio button. Specify the window title of the window containing the check box/radio button, and the object's caption. The caption of a check box or radio button is the text appearing next to it. The caption must be specified accurately with attention paid to case. Where a letter is underlined indicating a shortcut key, enter the '&' character before it.

window_title can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation : SBX
See also: GetCheckBox 255

Example

```
GetCheckBox>Internet Explorer Prop*,Never dial a &connection,res
```

© 2015 MJT Net Ltd

```
If>res=1
  SetCheckBox>Internet Explorer Prop*,Never dial a &connection,FALSE
Else
  SetCheckBox>Internet Explorer Prop*,Never dial a &connection,TRUE
Endif
```

4.27.10 SetControlText

SetControlText>WindowTitle,ClassName,Instance,NewText

Sets the text of the control specified by ClassName and Instance on the window specified by WindowTitle.

If WIN_USEHANDLE is set to 1 WindowTitle must be a window handle.

To determine class names of objects on windows use View System Windows 36. A window may contain several objects of the same class name. Instance is used to determine which instance of the class to use.
WindowTitle can end with an asterisk to indicate a substring match. WF_TYPE, WIN_USEHANDLE and WIN_REGEX directives are also accepted. See SetFocus 249 for a more detailed explanation of how the asterisk, WF_TYPE, WIN_USEHANDLE and WIN_REGEX can be used to define how the window is located.

Abbreviation: SCT
See also: GetControlText 256

Example:

```
SetControlText>Notepad*,Edit,1,Hello World
```

This will set the Notepad edit window to the text "Hello World".

4.27.11 SetObjectText

SetObjectText>handle,text

Attempts to set the published text property of an object given its handle.

See also: FindObject 255, SetControlText 261

Example:

```
//Set the text in Notepad's editor
GetWindowHandle>Untitled - Notepad,hWndParent
FindObject>hWndParent,Edit,,1,hWnd,X1,Y1,X2,Y2,result
SetObjectText>hWnd,This Text Was Autogenerated!
```

4.27.12 UIAccessibleList

UIAccessibleList>Window Title,Result

Not supported in Macro Scheduler Lite.

Returns the tree of visual accessible UI objects belonging to the specified window. Returns names,

© 2015 MJT Net Ltd

values (if any), positions and dimensions.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIA

See Also: UIClick 262, UIFocus 262, UIGetValue 263, UIPos 263, UISelect 263, UISetValue 264

4.27.13 UIClick

UIClick>Window Title,Object Name

Not supported in Macro Scheduler Lite.

Performs the default action of the specified accessible UI element for the given window. E.g. for a button this would usually be a click.

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible visible UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Note that to invoke menu items the parent window usually needs to be focused first (use SetFocus). Also sub-menu items are often not "accessible" and therefore cannot be accessed with the UI functions. Note that the item also has to be visible before it can be "clicked", so for e.g. sub menu items that are accessible you may first need to use UIClick on the parent menu item.

Abbreviation: UIC

See Also: UIAccessibleList 261, UIFocus 262, UIGetValue 263, UIPos 263, UISelect 263, UISetValue 264

4.27.14 UIFocus

UIFocus>Window Title,Object Name

Not supported in Macro Scheduler Lite.

Attempts to set keyboard focus to the specified accessible UI element of the given window.

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIF

See Also: UIClick 262, UIAccessibleList 261, UIGetValue 263, UIPos 263, UISelect 263, UISetValue 264

© 2015 MJT Net Ltd

4.27.15 UIGetValue

UIGetValue>Window Title,Object Name, Values,Positions

Not supported in Macro Scheduler Lite.

Returns a value or list of values, and positions and dimensions for objects matching Object Name on the given window.

If more than one object is found the values and positions will be delimited by the "|" character. Positions has the format (Xpos,YPos,Width,Height).

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIG

See Also: UIClick 262, UIFocus 262, UIAccessibleList 261, UIPos 263, UISelect 263, UISetValue 264

4.27.16 UIPos

UIAPos>Window Title,Object Name,Result

Not supported in Macro Scheduler Lite.

Returns the positions and dimensions for objects matching Object Name for the given window. If more than one match is found Result will be a list delimited by "|". The format for each object is Xpos,Ypos,Width,Height.

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIP

See Also: UIClick 262, UIFocus 262, UIGetValue 263, UIAccessibleList 261, UISelect 263, UISetValue 264

4.27.17 UISelect

UISelect>Window Title,Object Name

Not supported in Macro Scheduler Lite.

Attempts to select the specified object.

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible

© 2015 MJT Net Ltd

UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIZ

See Also: UIClick 262, UIFocus 262, UIGetValue 263, UIPos 263, UIAccessibleList 261, UISetValue 264

4.27.18 UISetValue

UISetValue>Window Title,Object Name,New Value

Not supported in Macro Scheduler Lite.

If possible will set the value of the specified UI automation element to the new value.

Use UIAccessibleList, the Find Object Wizard or View System Windows Tool to identify accessible UI elements.

Window title can be a full case sensitive title or case insensitive substring if followed by the '*' character. A handle can be specified if WIN_USEHANDLE is set to 1 or if WIN_REGEX is set to 1 a regular expression can be used. See SetFocus for more info.

Abbreviation: UIS

See Also: UIClick 262, UIFocus 262, UIGetValue 263, UIPos 263, UISelect 263, UIAccessibleList 261

4.28 Using Variables

Contents:

User Defined Variables 264
Arrays 265
Scope 266
Explicit Variable Resolution - VAREXPLICIT 267
Ignoring Spaces - IGNORESPACES 268
System Variables 269
Variable Explorer 272
Debugger - Watch List 22

4.28.1 User Defined Variables

As well as system variables, which are pre-set, variables can be assigned by script commands which return variables, and by the Let command which allows you to create variables at any time. All variables have global scope (are available to the entire script once created).

The Let command is used to assign a value to a variable like so :

```
Let>Name=Freddy Mercury
```

The first time a variable is assigned a value, that variable is created. Subsequently, it's value is modified.

© 2015 MJT Net Ltd

Variables can then be passed into other functions as one of the parameters. For example, assuming the above Let command has taken place, the following command will display a message box containing the words 'Freddy Mercury' :

```
Message>Name
```

If Name did not exist as a variable, the message box would simply display the word 'Name'.

Commands always check to see if a parameter passed to it is a variable. If a variable with that name exists, the value assigned to that variable is used, otherwise just that literal value is used.

All commands which accept a value can also accept a variable in place of that value.

For example, the following command would send the text 'Freddy Mercury' to the current window :

```
SendText>Name
```

In some cases a variable must be embedded within a string. This can be achieved using the % operator. In the following example a message is displayed saying 'Hello Freddy Mercury' :

```
Message>Hello %Name%
```

The system variable CRLF can be used to force a new line in a message box. Therefore, a list can be built up like so :

```
Message>1 : first line %CRLF%2 : second line %CRLF%3 : third line
```

It is possible to ask the user for a value, using the Input box :

```
Input>path,Please enter directory to create :
```

This would prompt the user to enter a directory name to create, which would then be assigned to the variable path. We could then use it in the CreateDir command as follows :

```
CreateDir>path
```

The If command can be used to check the value of a variable :

```
If>Age>15
  ..
Endif
```

4.28.2 Arrays

Arrays in MacroScript are essentially ordinary variables with a numeric suffix. The suffix format can take any format you prefer, although built in functions which return arrays use _n where n is the numeric identifier.

Array examples:

```
ArrayDim>Names,3
Let>Names_1=Fred
Let>Names_2=Sally
Let>Names_3=Geoff
```

© 2015 MJT Net Ltd

```
Let>k=2
Message>Names_%k%
Let>Names_%k%=Louise
```

Arrays created by MacroScript functions are displayed in the debugger watch list as <array> and can be expanded by double clicking to reveal the array contents. This avoids the watch list being cluttered up with all array elements and can speed up debugging scripts with large arrays. To get the same behaviour for user defined arrays use the ArrayDim command to create an empty array.

Array functions:

ArrayDim 172
ArraySort 172
ArrayCount 171
DelArray 173

Using VBScript Arrays:

Use the VBEval command to get the value of a VBScript array variable into a MacroScript variable:

```
VBEval>MyArray(2,3),theValue
```

4.28.3 Scope

By default variable scope is global. That is, any variable created in your script will be available anywhere else in your script, including in any subroutines. Any variables created within a subroutine will remain available to the rest of the script.

A word or two should be said about other scripts called or included by your script. The Macro 204 command executes another macro. That macro is kept separate from the calling macro. In this instance variables are not shared, so scope could be said to be local. Using the Macro 204 command is like calling a separate process via the command line. Variables can be passed into the called macro via command line parameters, but script variables within the code are not shared. In contrast the Include 202 statement, rather than "call" the included macro file, literally includes the code from that file within the main script. Include 202 is rather like copying and pasting the code from the Include file into the main script file at run time. Therefore in this instance variables are shared as the Included script becomes part of the main script.

Local Scope

It is possible to force new variables to have only local scope by setting LOCALVARS to 1. After setting LOCALVARS to 1 any variable creations or assignations will have local scope. I.e. if LOCALVARS is 1 and a new variable is created within a subroutine that variable will only have scope within that subroutine. It will no longer be available to the rest of the script once that subroutine has terminated.

With local scope enabled subroutines can still see any "global" variables. I.e. variables created in the main script or parent subroutine will be available to the subroutine. But attempting to modify an existing variable while LOCALVARS=1 will create a local version of that variable if it doesn't already exist locally. In other words all variable assignations or creations will have local scope.

Therefore if LOCALVARS has been set to 1 and you need to modify a lower level variable you will need to set LOCALVARS to 0 first.

Note that variable look-ups are recursive down through the variable table hierarchy so when retrieving values a match will occur against a parent level variable first. In other words variable look-ups "drill

© 2015 MJT Net Ltd

down" through the variable table hierarchy until a match is found. E.g. if we are at level 2 and there is a variable in the root table (level 0) and another with the same name at the parent level 1, then the value of the parent subroutine's variable will be returned (level 1). If you want to override this behaviour to specifically force only level 0 look up, set LOCALVARS to 2.

Example:

Watch what happens when you step through this script line by line.

```
Let>globalA=apple
Let>kk=2
Let>file_1=fred
Let>file_2=sally
Let>file_3=alan
ARC>file,count
ARS>file

GoSub>NewSub

MessageModal>%file_1% %globalA%

SRT>NewSub
    /*
    Set LOCALVARS to 1 to force all variable assignations and creations to have local scope.
    You can still retrieve global variables.  But attempting to modify them while LOCALVARS=1
    will cause creation of a local variable with same name.
    So to modify global variables, set LOCALVARS=0
    Setting LOCALVARS to 0 (default) means variables will be created/modified in the global tab
    */
    Let>LOCALVARS=1

    Let>kk=0

    MessageModal>globalA

    GetFileList>%USERDOCUMENTS_DIR%\*.*,filelist,;
    Separate>filelist,;,file

    Repeat>kk
      Let>kk=kk+1
      Let>Item=file_%kk%
    Until>kk=file_count

    //here we want to set the value of global variable globalA
    Let>LOCALVARS=0
    Let>globalA=pear

END>NewSub
```

4.28.4 Explicit Variable Resolution - VAREXPLICIT

You can tell Macro Scheduler only to resolve variables that are contained within % symbols and to leave anything else as literals by setting the VAREXPLICIT variable to 1:

```
Let>VAREXPLICIT=1
```

The default is 0 and will cause default behaviour as described above - i.e. Macro Scheduler will look at any parameter to see if it is a variable. If a variable with that name exists, the value assigned to

© 2015 MJT Net Ltd

that variable is used, otherwise just that literal value is used. Setting VAREXPLICIT to 1 stops this behaviour and tells Macro Scheduler only to resolve a variable if it is contained within % symbols. E.g.

```
Let>VAREXPLICIT=1
Let>Name=Fred
Send>Name
Send>%Name%
```

In this case the first Send> command will send the literal 'Name' and the second would send the value 'Fred'. In the following example, which is the default behaviour, both Send> commands would send the value 'Fred':

```
Let>VAREXPLICIT=0
Let>Name=Fred
Send>Name
Send>%Name%
```

4.28.5 Ignoring Spaces - IGNORESPACES

By default, spaces are seen as regular characters and are included in variable assignments. E.g. The following line..

```
Let>a = 5
```

.. would create a variable called "a " with the value " 5".

Normally, therefore, you should use:

```
Let>a=5
```

But experienced programmers are used to the language ignoring spaces. This can be achieved by setting IGNORESPACES to 1:

```
Let>IGNORESPACES=1
Let>a = 5
```

This would set variable "a" to the value 5.

If spaces are ignored then strings that need preceding spaces will need to be delimited. To do this use {"string"}

Try stepping through the example below with the Debugger and Watch List open.

Example

```
Let>VAREXPLICIT=1
Let>IGNORESPACES=1

Let>a = 5
Let>b = 22

Input>c, Enter a number:, 0
If> %c% > 0
  Let> d = {%a% + %b% * %c%}
Else
  Let> d = {%a% * %b%}
```

© 2015 MJT Net Ltd

```
Endif
MessageModal> Answer: %d%

//With IGNORESPACES=1 do the following if you want leading/trailing spaces in a
string
Let>string_with_spaces = {"  I want preceding spaces  "}
MessageModal> %string_with_spaces%

//Or just switch IGNORESPACES off and on
Let>IGNORESPACES=0
MessageModal>  Need these leading spaces
Let>IGNORESPACES=1

//Inline Expressions:
MessageModal>{"  I want preceding spaces"}

//Sum:
MessageModal>{5 + 4}

//Embed a complex expression within the string as you would a variable:
MessageModal>W Pos: %{Pos("W","Hello World")}%
```

4.28.6 System Variables

Environment Specific Values:

OS_VER	Operating System
WIN_DIR	Windows Directory Path
SYS_DIR	Windows System Directory Path
SYS_NATIVE	Native System Directory Path Bypassing 64 Bit Redirection
TEMP_DIR	Windows Temp Directory Path
DESKTOP_DIR	User's Desktop Path
USERDOCUMENTS_DIR	User's Documents Path
PROGRAM_FILES	Program Files Folder (Will be x86 folder if running on x64)
PROGRAM_FILES_NATIVE	Native Program Files Folder (e.g. "Program Files" not (x86) on x64)
OS_PLATFORM	Operating System Platform - i.e. WIN32 or WIN64

The above variables store their values in upper case.

USER_NAME	Current Username
COMPUTER_NAME	Computer Name
MSCHED_VER	Macro Scheduler version number

Runtime Variables:

SCRIPT_DIR	Directory of running script
SCRIPT_FILE	Filename (full path) of running script. For compiled macro this will be the path of the .exe file
SCRIPT_NAME	Just the filename part of the running script/exe
BMP_DIR	Folder used by image recognition wizard whose full path is SCRIPT_DIR\SCRIPT_NAME. Use of BMP_DIR for storing needle image simplifies distribution of image recognition macros and runtime images.
COMMAND_LINE	Full command line string (path of running executable and parameters passed on the command line)
MACRO_PARMS	Parameters passed by Macro> command
CWD	Current Directory
_LINE_NUM	Stores the current line number being executed

Useful printable and non-printable characters:

© 2015 MJT Net Ltd

CR	Carriage Return
LF	Line Feed
CRLF	Carriage Return, Line Feed Combination (to force a new line)
TAB	Tab character
SPACE	Space character
NULLCHAR	Null Character - Chr(0)
DECIMAL_SEPARATOR	Sets/retrieves the character used as the decimal separator in floating point/currency numbers. Default is "."
THOUSAND_SEPARATOR	Sets/retrieves the character used as the thousand separator in numbers. Default is ","
COMMA	,

Directives:

The following variables are directives used to alter the behaviour of various functions. Some have default values which are given below. Some have no default value which means they are "unset" and have no effect (the default behaviour of the function as described in the function topic will occur). Once set these variables remain set for the remainder of the script or until reset/changed. For more information about each one see the topic for the command they affect.

Variable Name	Default Value	Description
WW_TIMEOUT	0	Timeout value from WaitWindow Open 253 / Closed 251
WSI_TIMEOUT	0	Used to set a timeout value for WaitScreenImage 157
WST_TIMEOUT		Used to set a timeout value for WaitScreenText 225
RP_WAIT	0	Switches waiting for program termination on and off in RunProgram 194
RP_WINDOWMODE	1	Used to set window mode for RunProgram 194
RP_ADMIN		Set to 1 to make RunProgram 194 run as admin
RP_DISPLAYERROR		Allows RunProgram 194 error messages to be turned off/on
RP_WIN64PROCESS		Set to 1 to disable 64 bit redirection to make Run Program run a process stored in the 64 bit system folder rather than the 32 bit system folder
MSG_STAYONTOP	0	Used to set message to stay on top in Message 169 / MessageModal 170
MSG_CENTERED		Used to set message box to center
MSG_WIDTH	273	Used to set the width of the message box
MSG_HEIGHT	200	Used to set the height of the message box
MSG_XPOS	CENTER	Used to set the X position of the message box
MSG_YPOS	CENTER	Used to set the Y position of the message box
ASK_TIMEOUT		Used to set a timeout for Ask 168 dialog
SK_DELAY	0	Millisecond delay to pause between sending characters in SendText 162 or between repeated key presses sent with Press... 160
SK_IGNORECAPS		Set to 1 for Send 162 to ignore caps lock and send characters as entered
SK_LEGACY		If Press/Release commands fail in Citrix/DOS, set this to 1 first.
PRESS_ALLOWVARS		Set to 1 to make Press/Release commands accept a variable for the key name
RND_SEED	Random	Allows a seed to be set for the Random 180 command
CF_OVERWRITE	0	Allows changing CopyFile 118 to overwrite or rename on collision
FTP_STATUS		Used to switch off/on the FTP status window
FTP_PASSIVE		Used to toggle FTP passive mode
FTP_TIMEOUT		Timeout value for FTPGetFile 132 / FTPPutFile 134 / FTPGetDirList 131
FTP_USETLS		Set type of TLS/SSL connection to use
SSL_CERT		Specify SSL certificate file
SSL_ROOT_CERT		Specify SSL root certificate file
SSL_KEY		Specify SSL key file
SENDMAIL_STATUS		Used to switch off/on the SMTPSendMail 143 status window
SMTP_AUTH		Used to set SMTP authentication on or off
SMTP_USERID		Used for SMTP authentication
SMTP_PASSWORD		Used for SMTP authentication
SMTP_RECEIPT		Used to enable SMTP return receipt
SMTP_PORT		Used to optionally set the port of the SMTP server
SMTP_CCLIST		Set CC recipients for SMTPSendMail 143
SMTP_BCCLIST		Set BCC recipients for SMTPSendMail 143
SMTP_TIMEOUT		Timeout for SMTP connection (milliseconds)
SMTP_HTMLBODY		Set to 1 to send body as HTML in SMTPSendMail 143

© 2015 MJT Net Ltd

POP3_STATUS		Used to switch off/on the RetrievePOP3 [142] status window
POP3_PORT		Used to set the POP3 port (defaults to 110)
POP3_TIMEOUT		Used to set a POP3 timeout
POP3_DELETE		Used to tell RetrievePOP3 [142] to delete messages from the server
POP3_MSGSIZELIMIT		POP3 message size limit - RetrievePOP3 ignores messages larger
REG_INTASSTR		Used to set RegistryWriteKey to write integers as strings/integers
WF_TYPE	1	Used to set the type of window the window ing functions should affect
VBS_TIMEOUT		Used to set the timeout for VBScript code
GAW_TYPE	0	Used for GetActiveWindow [242] to set for active, or active child window .
INPUT_PASSWORD		Used to mask data entry in Input [168].
MF_RENAME		Set to 1 to cause MoveFile [126] to rename instead of physically move files.
APP_TITLE		Sets the application title of compiled macros.
WIN_USEHANDLE		Set to 1 to set all window functions to use/return window handles instead of window titles. Set to 0 to set back to window titles.
WIN_REGEX		Set to 1 to use regular expressions instead of plain text for window title matching in all window functions. Default is 0 (off).
WIN_SLEEP		Set to 1 to decrease CPU usage for (but slow down) WaitWindow Open/Closed and OnEvent WINDOW_OPEN/NOTOPEN.
HTTP_TIMEOUT		Optional timeout value for HTTPRequest [137]
HTTP_SSL		Specify whether to use SSL or not for HTTPRequest [137]
STEP_DELAY		Sets a millisecond delay between each script line
CRYPT_LEVEL		Sets the encryption strength for Crypt [212] command
GFL_TYPE		Set to 1 to make GetFileList return directories only
FIP_SCANPIXELS	ALL	Used to alter the number of random pixels FindImagePos [152] scans
WLN_NOCRLF		Set to 1 to make WriteLn omit line break.
WLN_ENCODING		Set to UNICODE to make WriteLn output Unicode (anything else/default is ANSI)
_DUMP_VARS		Set to 1 to dump all variable names and values to log file.
_WRITE_LOG_FILE		Set to 0 to temporarily disable logging. Set to 1 to enable.
VAREXPLICIT	0	Sets variable resolution method - see Using Variables [9].
IGNORESPACES	0	Set script to ignore leading and trailing spaces. See Using Variables [9].
IGNOREERRORS		Set to 1 to ignore script error messages. Does not apply to VBScript errors (use on error next etc for VBScript errors). Set back to 0 for default behaviour. See Error Handling [272].
LOCALVARS	0	Set to 1 to give variable assignations local scope. See Scope [266].
DISABLE_COMPLEX_EXPRESSIONS		Set to 1 to disable complex expressions [54]. Complex expressions will be seen as literal strings/regular variables.
CF_OVERWRITE	0	Set to 1 to force CopyFile [116], MoveFile [126] and RenameFile [128] to overwrite an existing target.

Result Variables:

Some functions set special result variables rather than use parameter variables passed to the function. This is mostly for backward compatibility reasons.

WW_RESULT	Result from WaitWindow Open [253] / Closed [251]
WCC_RESULT	Result from WaitCursorChanged [181]
WSI_TIMEDOUT	Indicates whether or not WaitScreenImage [157] timed out
WSI_BITMAP	The filename of the bitmap file which was not found when WaitScreenImage timed out
DDE_TIMEDOUT	Timeout value from DDERequest [70]
RP_RESULT	Used to store return code from RunProgram [194]
FTP_RESULT	Result of FTP [130] commands
SMTP_RESULT	Result of SMTPSendMail [143] command.
POP3_RESULT	Stores the result of the RetrievePOP3 [142] command
MACRO_RESULT	Used to set/retrieve the result of a called macro.
CF_RESULT	False if a CopyFile [116], MoveFile [126] or RenameFile [128] command was aborted
CF_RESULT_CODE	Numeric result code of a CopyFile [116], MoveFile [126] or RenameFile [128] command (0 is success).
POP3_MSGFILES	A semicolon delimited list of files downloaded by RetrievePOP3

© 2015 MJT Net Ltd

4.28.7 Variable Explorer

The variable explorer is activated from the Tools menu of the Script Editor, or by typing CTRL-ALT-V.

The variable explorer lists all the user defined variables created by the script and shows you where they are created and modified. Double click on a variable name header to insert it into the script. Double click on a line item to locate the line in the editor.

4.29 Error Handling

Error dialogs can be suppressed by setting IGNOREERRORS to 1:

```
Let>IGNOREERRORS=1
```

To retrieve the last error use the LAST_ERROR variable.

A custom error subroutine can be called when an error occurs by setting ONERROR:

```
Let>ONERROR=subroutine_name
```

E.g.

```
Let>ONERROR=MyErrorHandler

..
..

SRT>MyErrorHandler
  TimeStamp>%SCRIPT_DIR%\error.log,LAST_ERROR
END>MyErrorHandler
```

© 2015 MJT Net Ltd

Index

- % -

% 9

- * -

*/ 208

- / -

/* 208
// 208

- _ -

_DUMP_VARS 269

- A -

About 39
Abs 55
Add 189
add strings 212
Add Worksheet 109
AddDialogHandler 106
AddTrayHandler 170
AddTrayIcon 171
AES 210
AESEncrypt 210
App 115
APP_TITLE 269
Append Files 115
AppendFile 115
application 241
ArcTan 55
Arithmetic Functions 55
Arithmetic Operators 54
Array 171, 172
ArrayCount 171
ArrayDim 172
Arrays 9, 219, 265
ArraySort 172
ASC 158
Ascii 158
Ask 168
ASS 199
Assigned 199

- B -

Base64 211
Binary Data 174
Binary File 174
BLO 174
Block Keyboard 174
Block Mouse 174
BlockInput 174
Bookmarks 12
Boxer 37
branch 165, 200, 202
button 259

- C -

CAP 158
Caps Lock 158
CapsOff 158
CapsOn 158
Carriage Return 269
CBX 255
CDG 102
Cell 111, 114
CF_OVERWRITE 269
CHA 116
changed 251
ChangeDirectory 116
checkbox 255, 260
Class Name 244
click 184, 185, 186, 187, 188
clipboard 58
CLO 241
close 113, 241
closed 251
CloseDialog 102
CloseWindow 241
Code Builder 10
Code Folding 12
Code Snippets 11
COF 158
Color 103, 156, 181
ColorToRGB 172
Command Line 26
Command Locator 10
COMMAND_LINE 269
Commands 10
comment 208
comments 208
CompareBitmaps 152
compile 34
Complex Expressions 54, 57
COMPUTER_NAME 269
CON 212
ConCat 212
concatinate 212
Conditional Functions 57
Connect Database 60
Contents 38
Control 222, 256, 261
COP 116
copy 116
CopyFile 116
CopyFolder 116
Cos 55
COU 118
count 117, 118
CountDirs 117
CountFiles 118
CR 269
CRE 118
create 4
create directory 118
Create Exe 34
CreateDir 118
creating 4
CRLF 269
Cry 212
Crypt 212
CRYPT_LEVEL 269
CSV 118
CSVFileToArray 118
cursor 181, 184, 186
CWD 269

- D -

DAT 64
Database 59, 60, 61
date 62, 63, 64, 65, 67, 69, 121
DateAdd 62
DateDiff 63
DateLocal 63
DatePart 63

© 2015 MJT Net Ltd

Dates 62, 63
DateStamp 64
Day 64
DayOfWeek 64
DBClose 59
DBConnect 60
DBExec 60
DBQuery 61
DDE 70
DDE_TIMEDOUT 269
DDEPoke 70
DDERequest 70
Debugger 22
Decrypt 210, 212
DEL 119, 120
DelArray 173
delay 209
delete 32, 119, 120
Delete Array 173
Delete Column 110
Delete Folder 135
Delete Row 110
Delete Variable 173
Delete Worksheet 111
DeleteFile 119, 120
DeleteFolder 119, 120
DelTrayIcon 173
DelVariable 173
Desktop 22
Desktop Shortcut 36
Details 34
Dialog 102, 103, 104, 105, 106
Dialog Events 106
Dialogs 24, 102, 103, 105, 168
directories 117
Directory 122, 130, 131, 136, 147
DLL 175, 177
DOW 64
DPK 70
DRQ 70

- E -

EasyPatterns 218
EDI 120
edit 4
EditIniFile 120
Editor 37
Elapsed 69
Email 142, 143
Encode 211
Encrypt 210, 212
Encryption 18, 210, 212
End 164, 196, 197, 198
EndDialog 102
EndWhile 199, 210
Environment Variables 174, 181
Escape 211
Event Handler 204
Excel 109, 110, 111, 112, 113, 114
EXE 194
execute 194
ExecuteFile 194
Executing 21
exists 147, 148
Exit 32, 199
Exp 55
explicit 9
Explicit Variable Resolution 267
ExportData 174
ExtractFileExt 213
ExtractFileName 213
ExtractFilePath 213

- F -

Favorites 10
FDF 130
FDT 121
FGD 131
FGF 132
file 116, 119, 120, 121, 126, 148
File Event 16
File Extension 213
File I/O 127, 128
File System Redirection 182, 183
file transfer protocol 130, 131, 132, 133, 134, 136
FileDate 121
Filename 213
files 118, 122
FileSize 121
FileTime 121
FIN 242
find 242
FindImagePos 152
FindObject 255
FindWindowWithText 242
FIP_SCANPIXELS 269
FMD 133
focus 104, 249
focused 252
folder 116, 119, 120
Folder Event 16
folders 117
Font 33, 104
Format 214
Forms 102, 105
FPF 134
Frac 55
FRF 136
FSZ 121
ftp 130, 131, 132, 133, 134, 135, 136
FTP_PASSIVE 269
FTP_RESULT 269
FTP_STATUS 269
FTP_TIMEOUT 269
FTPDelFile 130
FTPGetDirList 131
FTPGetFile 132
FTPMakeDir 133
FTPPutFile 134
FTPRemoveDir 135
FTPRenameFile 136
Functions 164, 196, 197, 198

- G -

GAW 242
GAW_TYPE 269
GCP 184
GCT 222, 256
GDA 103
GDT 65
Get Text 222, 223, 224, 225
get title 242
GetActiveWindow 242
GetCaretPos 183
GetCheckBox 255

© 2015 MJT Net Ltd

GetClipBoard 58
GetControlText 222, 256
GetCursorPos 184
GetDate 65
GetDialogAction 103
GetDialogProperty 105
GetEnvVar 174
GetFileList 122
GetFocusedObject 256
GetListItem 257
GetNewestFile 122
GetOldestFile 123
GetPixelColor 154
GetRectCheckSum 155
GetScreenRes 155
GetTextAtPoint 222
GetTextInit 223
GetTextInRect 223
GetTextPos 224
GetTextReset 224
GetTime 66
GetTreeNode 258
GetWindowChildList 243
GetWindowHandle 243
GetWindowList 243
GetWindowNames 244
GetWindowParent 244
GetWindowPos 245
GetWindowProcess 245
GetWindowSize 246
GetWindowText 224, 246
GetWindowTextEx 225
GEV 174
GFL 122
GFL_TYPE 269
Gosub 164, 196, 197, 198
Goto 165, 200
GPC 154
GPW 245
Graphics 152, 155, 157
GRC 155
Grid Lines 34
group 19
groups 19
GTM 66
GTP 183
GWP 245
GWT 224, 246

- H -

Handle 243
Hash 215
Help 28
Hide 32
Hot Keys 19
hour 66
HTMLDecode 215
HTMLEncode 215
HTT 137
HTTP 137
HTTP_TIMEOUT 269
HTTPRequest 137

- I -

Idle 209
IEClickTag 229
IEContainsText 230
IECreate 230
IEExtractTable 231
IEExtractTableByName 231
IEExtractTag 232
IEExtractTagByAttrib 232
IEExtractTagByName 233
IEFormFill 233
IEFormSubmit 234
IEGetFromURL 234
IEGetHWND 235
IEGetTagPos 235
IEGetTags 139
IEGetTagsByAttrib 140
IEGetURL 235
IEGoBack 236
IEGoForward 236
IENavigate 236
IEOnDownload 237
IEQuit 237
IERefresh 238
IESetTimeout 238
IEShowIE 238
IETagEvent 140
IETagEventByAttrib 141
IEWait 142, 239
IEWaitDocumentComplete 239
IEWaitFileDownload 239
IEWaitForText 240
IEWaitNew 241
IEWaitTimeout 240
IF 146
IFC 148
IFD 147
IfDirExists 147
IFE 148
IfFileChanged 148
IfFileExists 148
IfNot 149
IfNotDirExists 125, 149
IfNotFileChanged 125, 150
IfNotFileExists 126, 150
IfNotWindowOpen 151, 248
IFW 151
IfWindowOpen 151
IGNOREERRORS 269
IGNORESPACES 269
Ignoring Spaces 268
Image Recognition 152, 155, 157
Include 202
IncludeFromVar 202
ini files 120, 127
INP 168
Input 168
INPUT_PASSWORD 269
Int 55
Internet 138
Internet Explorer 139, 140, 141
IsConnectedToInternet 138

- J -

JSON 215
JSONParse 215
JSONPath 215

- K -

keystroke 159, 160, 161, 162
KillProcess 175

© 2015 MJT Net Ltd

- L -

Label 165, 202
LabelToVar 216
Large Buttons 34
LCL 184
LClick 184
LDB 184
LDblClick 184
LDO 185
LDown 185
LEN 217
Length 217
Let 190, 203
LF 269
LFR 175
LIB 175, 177
LibFree 175
LibFunc 175
LibFuncW 177
LibLoad 177
Line Feed 269
List 34
List Box 257
Listing 130, 131, 136
ListView 257
LLD 177
Ln 55
lock 17
Log on 17
logging 18, 69
Logical Operators 55
Loop 165, 166, 167, 199, 200, 202, 208, 209, 210
Loops 165, 166, 167, 199, 200, 202, 208, 209, 210
Lowercase 217
LTrim 217
LUp 185

- M -

MAC 204
Macro 113, 204
Macro Properties 31
MACRO_PARMS 269
MACRO_RESULT 269
MCL 185
MClick 185
MD5 215
MDB 185
MDblClick 185
MDL 170
MDO 186
MDown 186
Menu 3
Menu Commands 3
Menu Options 3
Menus 260
Message 169
message box 169, 170
MessageModal 170
MF_RENAME 269
MID 217
MidStr 217
Min 66
minutes 66
MMR 186
MNU 260
ModTrayIcon 178
Mon 67
Month 67
MOU 186
mouse 184, 185, 186, 187, 188, 259
MouseMove 186
MouseMoveRel 186
MouseOver 187
MOV 126
MoveFile 126
MoveWindow 248
MSCHED_VER 269
MSG 169
MSG_CENTERED 269
MSG_HEIGHT 269
MSG_STAYONTOP 269
MSG_WIDTH 269
MSG_XPOS 269
MSG_YPOS 269
MUp 187
MVR 187
MVW 248

- N -

new 4
New Macro 31
New Workbook 109
NOF 159
NON 159
Num Lock 159
NumOff 159
NumOn 159

- O -

Object 105, 106
ObjectSendKeys 159
ObjectSendText 159
OnEvent 204
open 112, 151, 194, 253
Options 37
Organising 10
OS_VER 269

- P -

parameters 9
Path 213
pause 209
Pi 55
Pixel 156
PLA 178
Playing 21
PlayWav 178
POP3_DELETE 269
POP3_MSGFILES 269
POP3_MSGSIZELIMIT 269
POP3_PORT 269
POP3_RESULT 269
POP3_STATUS 269
POP3_TIMEOUT 269
PopupMenu 178
POS 218
Position 218
Power 55
Press 160
PrintDialog 103
Process 179, 182
Process ID 245
Process Name 245
ProcessExists 179
Property 105, 106

© 2015 MJT Net Ltd

PUS 259
push 259
PushButton 259
PutClipBoard 58
PyExec 179
Python 179
- Q -
quick launch 19
Quit 113
- R -
RAN 180
Random 180
RCL 188
RClick 188
RDA 103
RDB 188
RDblClick 188
RDK 192
RDO 188
RDown 188
RDV 192
REA 127
ReadFile 127
ReadIniFile 127
ReadLn 127
reboot 250
Record 38
Recorder 20
Recording 20
REG_INTASSTR 269
RegEx 218
registry 192, 193
RegistryDelKey 192
RegistryDelVal 192
RegistryEnumKeys 192
RegistryEnumVals 193
RegistryReadKey 193
RegistryWriteKey 193
Regular Expressions 218
Relational Operators 55
relative 186
Release 161
Remark 208
Rename 32, 128
RenameFile 128
Repeat 166, 167, 208, 209
ResetDialogAction 103
ResizeWindow 249
RET 142
RetrievePOP3 142
RFL 127
RGB 181
RLN 127
RND_SEED 269
Round 55
RP_ADMIN 269
RP_DISPLAYERROR 269
RP_RESULT 269
RP_WAIT 269
RP_WINDOWMODE 269
RPL 219
RRK 193
RSW 249
RTrim 219
run 194
Run Now 38
Running 21
RunProgram 194
Runtime statistics 20
RUp 188
RWK 193
- S -
Save 113
SBX 260
Scheduler 12
Scheduling 12, 15
Scope 266
SCP 156
Screen Capture 156
Screen Image 157
Screen Resolution 155
ScreenCapture 156
screensaver 17
script 4
SCRIPT_DIR 269
SCRIPT_FILE 269
Scroll Lock 162
ScrollOff 162
ScrollOn 162
SCT 261
SDW 250
Sec 67
seconds 67
SelectMenu 260
SEN 162
SENDMAIL_STATUS 269
SendText 162
SEP 219
Separate 219
SET 249
SetCheckBox 260
SetControlText 261
SetDialogObjectColor 103
SetDialogObjectFocus 104
SetDialogObjectFont 104
SetDialogObjectVisible 105
SetDialogProperty 106
SetEnvVar 181
SetFocus 249
SetObjectText 261
SetRoundMode 191
SEV 181
SHA1 215
SHA256 215
Shortcut 22
Shortcuts 19
Show 105
shutdown 250
ShutDownWindows 250
Sin 55
size 121
SK_DELAY 269
SK_IGNORECAPS 269
SK_LEGACY 269
SkipLabel 198
SMT 143
SMTP_AUTH 269
SMTP_BCCLIST 269
SMTP_CCLIST 269
SMTP_PASSWORD 269
SMTP_PORT 269
SMTP_RECEIPT 269

© 2015 MJT Net Ltd

SMTP_RESULT 269
SMTP_TIMEOUT 269
SMTP_USERID 269
SMTPSendMail 143
SOF 162
SON 162
Sort 33
sound 178
SOWrite 59
SOWriteLn 59
SPACE 269
SQL 60, 61
Sqr 55
SRT 164, 196, 197, 198
Startup 15
STDOUT 59
STEP_DELAY 269
Stop 38
string 217, 218
String Functions 211, 212, 213, 216, 217, 218, 219, 220
String Operators 55
StringReplace 219
Strings 219
Sub 191
Subroutines 164, 196, 197, 198
substring 217, 218
subtract 191
Support 28
Switches 26
SYS_DIR 269
System Variables 269

- T -

TAB 269
TBR 189
Telnet 144, 145
TelnetClose 144
TelnetConnect 130, 144
TelnetSend 145
TelnetWaitFor 145
TEMP_DIR 269
Terminate 199
text 242
Text Capture 222, 223, 224, 225
Text Cursor 183
Text Editor 37
TIM 69
time 66, 67, 68, 121
TimeAdd 67
TimeDiff 68
TimeLocal 68
TimePart 68
Timer 69
TimeStamp 69
Title 244
Toolbar 4, 34, 189
Tray Icon 178
TreeView 258
Triggers 16
Trim 220
Trunc 55

- U -

UIAccessibleList 261
UIClick 262
UIFocus 262
UIGetValue 263
UIPos 263
UISelect 263
UISetValue 264
unlock 17
Until 166, 167, 208, 209
Uppercase 220
User Defined Variables 264
USER_NAME 269

- V -

VAREXPLICIT 9, 269
variable 190, 203
Variable Explorer 272
Variable Scope 266
Variables 9, 272
VBE 226
VBEND 226
VBEval 226
VBR 228
VBRun 228
VBS_TIMEOUT 269
VBScript 226, 228
VBSTART 228
View Log File 36
View System Windows 36
Visible 105

- W -

Wait 209
wait for key press 163
WaitClipBoard 58
WaitCursorChanged 181
WaitKeyDown 163
WaitPixelColor 156
WaitProcessExists 182
WaitProcessTerminated 182
WaitReady 196, 209
WaitRectChanged 157
WaitScreenImage 157
WaitScreenText 225
WaitWindow Changed 251
WaitWindow Closed 251
WaitWindow Focused 252
WaitWindow Open 253
wav 178
WCC 181
WCC_RESULT 269
WEB 137
Web Scraping 139, 140
WebRecorder 25
WF_TYPE 269
While 199, 210
WIN 254
WIN_DIR 269
WIN_USEHANDLE 269
window 151, 241, 242, 249, 251, 252, 253
Window Event 16
Window List 243
window position 245
Window Size 246
Window Text 224, 246
Window Action 254
WKD 163
WLN 128
Workbook 109, 112, 113
Worksheet 109, 111
Worksheet Dimensions 112
Wow64DisableRedirection 182
Wow64EnableRedirection 183

© 2015 MJT Net Ltd

WPC 156
WRC 157
WRD 209
WriteLn 128
WSI_TIMEOUT 269
WST_TIMEOUT 269
WW_RESULT 269
WW_TIMEOUT 269
WWC 251
WWF 252
WWO 253
WWX 251

- X -

XLAddSheet 109
XLCreate 109, 110
XLDelCol 110
XLDelRow 110
XLDelSheet 111
XLGetCell 111
XLGetSheetDims 112
XLOpen 112
XLQuit 113
XLRun 113
XLSave 113
XLSetCell 114
XLSheetToArray 114
XML 220
XMLParse 220
XMLPath 220

- Y -

Year 69

- Z -

Zip 129
ZipAddFiles 129
ZipExtractFiles 129

© 2015 MJT Net Ltd